Programare funcţională – Laboratorul 1

Common Lisp - Introducere

Isabela Drămnesc

February 20, 2012

1 Concepte

- Programare funcţională
- Common Lisp
- Interpretor Lisp
- read-eval-print
- Atomi, Liste
- Operaţii pe liste

2 Link-uri utile

- Resurse pentru laborator.
- tutorial Common Lisp
- inca un tutorial Common Lisp
- descărcare CLISP

3 Recursivitate

Exemple:
1) înmulţire

$$x * y = f[x, y] = \begin{cases} 0 & \text{daca } x = 0; \\ f[x-1, y] + y & \text{daca } x \neq 0. \end{cases}$$

2) ridicare la putere

$$x^y = f[x, y] = \begin{cases} 1 & \text{daca } y = 0; \\ f[x, y-1] * x & \text{daca } y \neq 0. \end{cases}$$

3) factorial

$$n! = f[n] = \begin{cases} 1 & \text{daca } n = 0; \\ f[n-1] * n & \text{altfel.} \end{cases}$$

4) Scrieţi o funcţie recursivă pentru aflarea lungimii unei liste!

4 Introducere in CLISP

La lansarea interpretorului este afişat un prompter (>), iar funcţionarea interpretorului se bazează pe repetarea ciclului de bază read-eval-print.

1. read: citeşte o expresie simbolică;
2. eval: evaluează expresia simbolică introdusă;
3. print: afişează rezultatul obţinut în urma evaluării expresiei.

4.1 Aritmetică

4.1.1 Tipuri de numere:

Common Lisp pune la dispozitie 4 tipuri distincte de numere: intregi, float, rationale si numere complexe.

- *Numar intreg* este scris ca un sir de cifre: 2012 ;
- *Numar float* este scris ca un sir de cifre cu zecimale: 292.51, sau in notatie stiintifics: 2.9251e1 ;
- *Numar rational* este scris ca o fractie de numere intregi: 3/4 ;
- *Numar complex* $a + bi$ este scris ca #c(a,b).

In Lisp, funcţiile $F[x, y]$ sunt definite ca: (F x y)
$x + y$ este defapt $+[x, y]$, scris ca (+ x y)
Exemplu: (+ 4 6)

```
> (+ 4 6)

> (+ 2 (* 3 4))

> 3.14

> (+ 3.14 2.71)

> (- 23 10)

> (- 10 23)

> (/ 30 3)

> (/ 25 3)

> (/ (float 25) (float 3))

> (/ (int 25) (int 3))

> abort

> (/ 3 6)
```

```
1/2         ;numar rational

> (/ 3 6.0)
0.5         ;numar de tip float

> (max 4 6 5)

> (max 4 6 5 10 9 8 4 90 54 78)

> (min 8 7 3)

> (min 4 6 5 10 9 8 2 90 54 78)

> (expt 5 2)

> (expt 10 4)

> (sqrt 25)

> (sqrt 25.0)

> (sqrt -25)

> (sqrt -25.5)

> (abs -5)

> (+ (* 2 3 5) (/ 8 2))

> pi

> ()
NIL         ;simbol special in Lisp pentru "no", "lista vida"

> t    ;simbol special in Lisp pentru adevar ("yes")
T

> "a string"

> 'la la'

> a

>'a

> (truncate 17.678)
; retureaza componentul intreg al unui numar real

> (round 17.678)
```

```
> (rem 14 5)

> (mod 14 5)

> (+ #c(1 -1) #c(2 1))
```

4.2 Apostrof '

```
> 3
3       ; un numar se evalueaza la numarul insusi

> "hello"
HELLO       ; un sir de caractere se evalueaza la el insusi

> (+ 2 3)
5           ; se aplica + la 2 si 3

> a
ERROR: variable A has no value
           ;cauta sa evalueze pe a
```

Pentru a stopa evaluarea se foloseşte apostrof:

```
> '3

> '(+ 2 3)

> 'a

> (eval '(+ 2 3))   ; eval forteaza evaluarea

> '(2 3 4)

> (+ 10 20 30 40 50)

> '(eval '(+ 3 4))

>''3
```

Ce se întâmplă dacă scriu (2 3 4) în Lisp?
Cum pot afişa lista (2 3 4) în Lisp?

4.3 Predicate predefinite

numere intregi:

- o secvenţă de cifre de la 0 la 9 (opţional cu semnul plus sau minus in faţă);

simboluri:

- orice secvenţă de caractere şi caractere speciale care nu sunt numere;

De exemplu: +9 este un număr întreg, pe când + este un simbol.
10-23 este tot un simbol.

```
> (numberp 2)
T

> (numberp 22222222222222222222222222222222222222222222222222)
T

> (numberp 'dog)
NIL

> (symbolp 'dog)
T

> (symbolp +)
NIL

> (symbolp '+)
T

> (symbolp '9)
NIL

> (atom 3)
T

> (atom 'dog)
T

> (atom '45_dog)
T

> (atom '(a b c))
NIL

> (stringp "a string")
T

> (stringp '(a string))
NIL
```

Predicatele: integerp, floatp, ratiop, complexp returneaza true pentru numere de tipul corespunzător.

4.4 Liste (CAR CDR CONS)

Reprezentarea internă a listelor este dată de o structură arborescentă.
Exemple (reprezentare structură pentru:)
1) (A B C)
2) (A (B C))

3) (3 R . T)
4) (NIL)
5) ((A (B C)) D ((E F) G) H)
Listele sunt reprezentate ca:

Head (Cap) şi

Tail (Coadă).

Capul este un element, iar coada este o listă.
In LISP sunt 3 operaţii fundamentale pe liste:

```
Head(a b c d)=a --un element
Tail(a b c d)=(b c d) -- o lista
Insert[a, (b c d)]=(a b c d)
```

- Head CAR
- Tail CDR
- Insert CONS

Construirea listelor utilizând:

- cons
- list
- append

cons:

```
> (cons 'a nil)

> (cons 'a 'b)
      ;reprezentare in celule

> (cons 1 2 nil)
ERROR          ; doar doua argumente poate avea cons

> (cons 32 (cons 25 (cons 48 nil)))

> (cons 'a (cons 'b (cons 'c 'd)))

> (cons 'a (cons 'b (cons 'c '(d))))
```

list:

```
> (list 'a)

> (list 'a 'b)
```

```
>(list 32 25 48)
>(list a b c)
>(list 'a 'b 'c)
```

append:

```
> (append '(a) '(b))
```

car, cdr, cons:

```
> (car '(a b c))
> (cdr '(a b c))
> (car (cdr '(a b c d)))
> (car (cdr (car '((a b) c d))))
> (cdr (car (cdr '(a (b c) d))))
> (cdr (cons 32 (cons 25 (cons 48 nil))))
> (car (cons 32 (cons 25 (cons 48 nil))))
> (cdr (cdr (cons 32 (cons 25 (cons 48 nil)))))
> (cdr (cdr (cdr (cons 32 (cons 25 (cons 48 nil))))))
> (cdr (cdr (cdr (cons 32 (list 32 25 48)))))
> (cdr (cdr (cdr (list 32 25 48))))
> (cddr '(astazi este soare))
> (caddr '(astazi este soare si cald))
> (cdr (car (cdr '(a (b c) d)))) ; echivalent cu (cdadr '(a (b c) d))
> (nthcdr 0 '(a b c d e)) ; aplica cdr de 0 ori
> (nthcdr 1 '(a b c d e)) ;aplica cdr o data
```

Alte exemple:

```
> (cons '+ '(2 3))
> (eval (cons '+ '(2 3)))
```

```
> (length '(1 2 d f))

> (reverse '(3 4 5 2))

> (append '(2 3) (reverse '(i z a)))

> (first '(s d r))

> (rest '(p o m))

> (last '(p o m))

> (member 'om '(un om citeste))

> (car (member 'sapte '(o saptamana are sapte zile)))

> (subst 'maine 'azi '(azi este marti))
```

4.5 Comenzi utile:

- exit sau quit – pentru a părăsi interpretorul.
- :h Help
- **trace.** – Urmăreşte interactiv fiecare pas al execuţiei.

4.6 Tema:

1. Pentru fiecare din următoarele expresii Lisp desenaţi celulele de reprezentare pentru structura cons şi scrieţi ce afişează fiecare din cele de mai jos:

```
> (cons 'the (cons 'cat (cons 'sat 'nil)))

> (cons 'a (cons 'b (cons '3 'd)))

> (cons (cons 'a (cons 'b 'nil)) (cons 'c (cons 'd 'nil)))

> (cons 'nil 'nil)
```

Rescrieţi cele de mai sus utilizând list!

2. Desenaţi celulele de reprezentare şi scrieţi sintaza Lisp corespunzătoare utilizând cons şi list pentru fiecare din următoarele :

(THE BIG DOG)

(THE (BIG DOG))

((THE (BIG DOG)) BIT HIM)

(A (B C . D) (HELLO TODAY) I AM HERE)

3. Utilizaţi car, cdr, şi combinaţii ale lor pentru a returna:

LISTA: (A (L K (P O)) I) returneaza: O si (O)

LISTA: (A ((L K) (P O)) I) returneaza: O si (K)

LISTA: (A (B C . D) (HELLO TODAY) I AM HERE) returneaza HELLO, apoi AM

Notă: Termen de realizare: laboratorul următor.