Programare Funcţională – Laborator 1

Introducere în DrRacket

Isabela Drămnesc

February 26, 2014

1 Concepte

- Programare Funcţională
- Interpretor DrRacket
- read-eval-print
- Atomi, Liste
- Predicate, funcţii predefinite
- Operaţii pe liste

2 Utile

- Laboratoarele de PF.
- O scurta introducere in DrRacket
- The Racket Guide
- Descărcare gratuită DrRacket

3 Introducere in DrRacket

Mecanismul se bazează pe ciclul read-eval-print.

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

```
; comentariu pe o singura linie
#| comentariu pe mai multe linii
   comentariu pe mai multe linii
|#
```

3.1 Tipuri de date Built-In

- Booleans
- Numbers
- Characters
- Stings
- Bytes and Byte Strings
- Symbols
- Keywords
- Pairs and Lists
- Vectors
- Hash Tables
- Boxes
- Void and Undefined

3.2 Aritmetică

3.2.1 Tipuri de numere:

În Racket un număr este fie exact fie inexact:

- Un număr exact poate fi:
  - *intreg*: -9 ; 2014 ; 88888888888888888888888888;
  - *rational* scris ca o fracţie de întregi: 3/4 ; -1/2; 77777777777777777777777/2;
  - *complex* (care are parte reală şi parte imaginară, iar partea imaginară nu este zero) $a + bi$ se scrie ca $a + bi$; $1/2 + 3/4i$.
- Un număr inexact poate fi:
  - *float*: 292.51, sau: 2.9251e1 ; 2.0 sau 3.14e+87
  - *complex* (cu partea reală şi imaginară float): 3.0 + 4.0i

3.2.2 Funcţii:

Funcţiile $F[x, y]$ sunt definite astfel: (F x y)
$x + y$ este defapt $+[x, y]$, scrisă ca (+ x y)
Exemple: (+ 4 6)

```
> (+ 4 6)

> (+ 2 (* 3 4))

> 3.14

> (+ 3.14 2.71)

> (- 23 10)

> (- 10 23)

> (/ 30 3)

> (/ 25 3)

> (/ 3 6)

> (/ 3 6.0)

> (max 4 6 5)

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

> (max 4 5)     ;exact

> (max 3.9 4) ; inexact

> (min 8 7 3)

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

> (expt 5 2)
; exponentiation

> (expt 10 4)

> (sqrt 25)
; square root

> (sqrt 25.0)

> (sqrt -25)

> (sqrt -25.5)

> (abs -5)

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

```
> (modulo 13 4)

> pi

> '()
empty        ; "empty list"

> #t    ;simbol special in Racket pentru adevarat
true

> #f    ;simbol special in Racket pentru fals
false

> "a string"

> 'la la'

> a

>'a

> (round 17.678)

> (floor 7.5)

> (ceiling 7.5)

> (+ 1+2i 3+4i)
```

3.3 Quote '

Este un operator special care are rolul de a stopa evaluarea.

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

> "hello"

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

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

Pentru a stopa evaluarea folosim operatorul quote:

```
> '3

> '(+ 2 3)
```

```
> 'a

> (eval '(+ 2 3))  ; eval forces the evaluation

> '(2 3 4)

> (+ 10 20 30 40 50)

> '(eval '(+ 3 4))

>''3
```

Ce se intampla daca scriem (2 3 4)?
Cum putem obtine lista cu elementele (2 3 4)?

3.4 Predicate predefinite

De exemplu: +9 este un numar intreg, + este simbol.
10-23 este tot un simbol.

```
> (number? 2)

> (number? 22222222222222222222222222222222222222222222222222)

> (number? 'dog)

> (symbol? 'dog)

> (symbol? +)

> (symbol? '+)

> (symbol? '9)

> (string? "a string")

> (string? '(a string))

> (boolean? #f)

> (boolean? 0)

> (boolean? '())
```

Alte predicate: integer?, real?, rational?, complex?

3.5 Liste (CAR CDR CONS)

Listele se reprezintă folosind celule cons.
Exemple:
1) (A B C)

2) (A (B C))
3) (3 R . T)
4) '()
5) ((A (B C)) D ((E F) G) H)
Listele sunt reprezentate prin: Head şi Tail.
Head-ul este un element, iar tail-ul este o listă.
In Racket există 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 folosind:

- cons
- list
- append

cons:

```
> (cons 'a '())

> (cons 'a 'b)
(A . B)          ;the representation of cells

> (pair? '(a . b)

> (cons 1 2 nil)
ERROR            ; we can use cons only with two arguments

> (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 '()))))
> (car (cons 32 (cons 25 (cons 48 '()))))
> (cdr (cdr (cons 32 (cons 25 (cons 48 '())))))
> (cdr (cdr (cdr (cons 32 (cons 25 (cons 48 '()))))))
> (cdr (cdr (cdr (cons 32 (list 32 25 48)))))
> (cdr (cdr (cdr (list 32 25 48))))
> (cddr '(today is sunny))
> (caddr '(today is sunny and warm))
> (cdr (car (cdr '(a (b c) d)))) ; equivalent with (cdadr '(a (b c) d))
```

Alte exemple:

```
> (cons '+ '(2 3))
> (eval (cons '+ '(2 3)))
> (length '(1 2 d f))
> (reverse '(3 4 5 2))
> (append '(2 3) (reverse '(i s a)))
```

```
> (first '(s d r))

> (rest '(p o m))

> (memq 'man '(a man is reading))

> (car (memq 'seven '(a week has seven days)))
```

3.6 Temă:

1. Pentru fiecare din expresiile următoare scrieţi rezultatul evaluării şi desenaţi celulele de reprezentare.

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

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

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

> (cons 'nil 'nil)
```

Rescrieţi cele de mai sus folosind list !

2. Desenaţi celulele de reprezentare pentru fiecare din listele următoare şi scrieţi sintaxa Racket corespunzătoare pentru a le obţine:

(THE BIG DOG)

(THE (BIG DOG))

((THE (BIG DOG)) BIT HIM)

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

3. Folosiţi car, cdr şi combinaţii pentru:

List de input este: (A (L K (P O)) I) returneaza: O respectiv (O)

List de input este: (A ((L K) (P O)) I) returneaza: O respectiv (K)

List de input este: (A (B C . D) (HELLO TODAY) I AM HERE) returneaza HELLO, apoi AM

Termen de finalizare temă: următorul laborator.