Functional Programming – Laboratory 1

Introduction in DrRacket

Isabela Drămnesc

February 25, 2014

1 Concepts

- Functional Programming
- DrRacket interpretor
- read-eval-print
- Atoms, Lists
- Lists operations

2 Useful Links

- Laboratories.
- A short introduction tutorial on DrRacket
- The Racket Guide
- Free download DrRacket

3 Introduction in DrRacket

When you start the interpreter, usually a prompt > is displayed to tell you that it's waiting for you to type something.

The mechanism is based on the loop read-eval-print.

1. read: reads a symbolic expression;
2. eval: evaluates the introduced symbolic expression;
3. print: displays the result obtained after evaluating the introduced expression.

```
; single line comment
#| multi-line comment
   multi-line comment
|#
```

3.1 Built-In Datatypes

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

3.2 Arithmetic

3.2.1 Types of numbers:

A Racket number is either exact or inexact:

- An exact number can be:
  - *An integer* is written as a string of digits: -9 ; 2014 ; 88888888888888888888888888;
  - *A rational* is written as a fraction of integers: 3/4 ; -1/2; 77777777777777777777/2;
  - *A complex* number (which has a real and an imaginary part, the imaginary part is not zero) $a + bi$ is written as $a + bi$ or $1/2 + 3/4i$.
- An inexact number can be:
  - *A floating-point* number is written as a string of digits containing a decimal point: 292.51, or in scientific notation: 2.9251e1 ; 2.0 or 3.14e+87
  - *A complex* number (with real and imaginary parts as floating point representations): 3.0 + 4.0i

3.2.2 Functions:

The functions $F[x, y]$ are defined as: (F x y)
$x + y$ is actually $+[x, y]$, written as (+ x y)
Example: (+ 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    ;a special symbol in Racket for truth
true

> #f    ;a special symbol in Racket for false
false

> "a string"

> 'la la'

> a

>'a

> (round 17.678)

> (floor 7.5)

> (ceiling 7.5)

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

3.3 Quote '

This is a special operator which has its own rule, namely "do nothing". The quote operator takes a single argument and returns its verbatim.

```
> 3
      ; a number evaluates to itself

> "hello"
      ; a string evaluates to itself

> (+ 2 3)
      ; applies + to 2 and to 3

> a
ERROR: variable A has no value
```

```
                ;he wants to evaluate a
```

In order to stop the evaluation we use the quote operator:

```
> '3

> '(+ 2 3)

> 'a

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

> '(2 3 4)

> (+ 10 20 30 40 50)

> '(eval '(+ 3 4))

>''3
```

What happens when we type (2 3 4)?
How can we print the list containing the elements (2 3 4)?

3.4 Predefined Predicates

integer numbers:

- a sequence from 0 to 9 (optional with the plus or the minus sign in front);

symbols:

- any sequence of characters and special characters which are not numbers;

For example: +9 is an integer number, but + is a symbol.
10-23 is also a symbol.

```
> (number? 2)

> (number? 22222222222222222222222222222222222222222222222)

> (number? 'dog)

> (symbol? 'dog)

> (symbol? +)

> (symbol? '+)

> (symbol? '9)

> (string? "a string")
```

```
> (string? '(a string))

> (boolean? #f)

> (boolean? 0)

> (boolean? '())
```

The predicates: integer?, real?, rational?, complex? return true for numbers of the corresponding types.

3.5 Lists (CAR CDR CONS)

We can represent the lists as trees of cons cells.

Examples (the box notation for each of the following printed representations of cons cells):

1) (A B C)
2) (A (B C))
3) (3 R . T)
4) '()
5) ((A (B C)) D ((E F) G) H)

Lists are represented as: Head and Tail.

The head is an element and the tail is a list.

In Racket there are 3 fundamental operations on lists:

```
Head(a b c d)=a --an element
Tail(a b c d)=(b c d) -- a list
Insert[a, (b c d)]=(a b c d)
```

- Head CAR
- Tail CDR
- Insert CONS

Constructing lists using:

- 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))
```

Other examples:

```
> (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 Homework:

1. For each of the following expressions draw the box notation for the cons structures it creates and write down the printed representation:

```
> (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)
```

Rewrite the expressions above by using list !

2. Draw the box notation for each of the following printed representations of cons cells and write the corresponding Racket syntax by using cons and list for each of the following:

(THE BIG DOG)

(THE (BIG DOG))

((THE (BIG DOG)) BIT HIM)

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

3. Use car, cdr, and combinations of it in order to return:

The input **list** is: (A (L K (P O)) I) returns: O **and** (O)

The input **list** is: (A ((L K) (P O)) I) returns: O **and** (K)

The input **list** is: (A (B C . D) (HELLO TODAY) I AM HERE) returns HELLO, then AM

Deadline: next laboratory.