Programare funcţională – Laboratorul 7

Macro-uri

1 Discutarea temei

2 Studiaţi următoarele exemple:

2.1 Macro dotimes

fara si cu destructuring lambda list:
-expandare, testare, modificare daca e necesar

```
(defmacro do-times (l &body corp)
        (let ((g (gensym)))
                `(do ((,(car l) 0 (+ ,(car l) 1))
                        (,g ,(cadr l)))
                        ((>= ,(car l) ,g) ,(caddr l))
                ,@corp)))

(defmacro do-times ((i n &optional r) &body corp)
        (let ((g (gensym)))
                `(do ((,i 0 (+ ,i 1))
                        (,g ,n))
                        ((>= ,i ,g) ,r)
                ,@corp)))
```

2.2 Macro ifnr

Testare, expandare, modificare dacă e necesar pentru a nu avea erori de scriere.
IFNR - un test numeric cu următoarea configuraţie:

```
(ifnr (<var><test>)
        (<ramura - >)
        (<ramura 0 >)
        (<ramura + >)
)
```

NU putem folosi o funcţie deoarece aceasta îşi evalueaza toate argumentele.

Condiţionala pe care ne-o propunem presupune evaluarea testului şi doar a uneia din cele trei forme. Deci folosim construcţia macro (care nu îşi evaluează toate argumentele).

```
> (defmacro ifnr (test ramura- ramura0 ramura+)
        (list 'let (list test)
```

```
        (list 'cond (append
        (list (list 'minusp (car test)))
        ramura-
        )
(append
        (list (list 'zerop (car test)))
                ramura0
)
(append '(t) ramura+ )
)
)
)
```

IFNR

```
> (ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
5
Ramura pozitiva
5


> (ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
-900
Ramura negativa
900


> (ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
0
Ramura 0
0


> (macroexpand '(ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
)
```

```
(LET ((X (READ)))
(COND ((MINUSP X) (PRINC "Ramura negativa") (- X))
          ((ZEROP X) (PRINC "Ramura 0") 0)
      (T (PRINC "Ramura pozitiva") X))) ;
T


; in exemplul anterior folosim backquote
;; Utilizand facilitatile ( ` , @ ) putem rescrie ifnr

(defmacro ifnr (test ramura- ramura0 ramura+)
        `(let ( (,(car test) ,(cadr test)))
        (cond ((minusp ,(car test))
                                    ,@ramura-)
                  ((zerop ,(car test))
                                    ,@ramura0)
                  (t ,@ramura+)
                )
        )
)

> (ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
5
Ramura pozitiva
5


> (ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
-900
Ramura negativa
900


> (ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
0
Ramura 0
0
```

```
;;; DESTRUCTURARE
;-------------------
;;;; si mai simplu

(defmacro ifnr ((var val) ramura- ramura0 ramura+)
`(let ((,var ,val))
        (cond ((minusp ,var) ,@ramura-)
                ((zerop ,var) ,@ramura0)
                (t ,@ramura+)
        )
        )
)
IFNR

> (macroexpand '(ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
))
(LET ((X (READ)))
        (COND ((MINUSP X) (PRINC "Ramura negativa") (- X))
                ((ZEROP X) (PRINC "Ramura 0") 0)
                (T (PRINC "Ramura pozitiva") X))) ;
T

> (ifnr (x (read))
        ((princ "Ramura negativa") (- x))
        ((princ "Ramura 0") 0)
        ((princ "Ramura pozitiva") x)
)
-980
Ramura negativa
980
```

3 Exercitii macro – expandare, apelare, testare

3.1 Suma

Definiţi un macro pentru calculul:

```
> (suma-macr 1 2 3 4 5)
15

> (suma-macr 10 20 30 40 50 10 -10 10)
160
```

3.2 Lungime

Definiţi un macro pentru calculul:

```
> (lungime-macr '(1 2 3 4 5))
5

> (lungime-macr '(10 20 30 40 50 10 a -10 b 10))
10
```

3.3 Sortare

Implementaţi algoritmul de sortare prin intercalare pentru o listă de întregi în Lisp – informal, acesta poate fi formulat astfel:
Fiind dată o listă, de împarte în două părţi de mărimi egale. Se sortează cele două părţi, iar rezultatele sortate se combină prin intercalare astfel ca ordinea elementelor să se păstreze. (NU folosiţi sort)

1. versiunea recursivă;
2. versiunea final recursivă;
3. versiunea iterativă;
4. macro.

```
> (inter-sort '(1 -3 2 0))
(-3 0 1 2)

> (inter-sort '(1 a 5 6 -8))
"lista nu contine doar numere, deci nu stim sa o sortam"
```

3.4 Înmulţirea "a la russe"

Se scrie x alături de y pe aceeaşi linie. Se împarte succesiv x la 2, se înmulţeşte y cu 2, procedeul continuă până când pe coloana lui x se obţine valoarea 1. Se adună toate valorile de pe coloana lui y care corespund valorilor impare de pe coloana lui x.

1. varianta iterativă;
2. varianta folosind macro.

4 Tema

Studiaţi următoarele:

```
(defmacro -prog1 (arg1 &rest args)
  (let ((g (gensym)))
       `(let ((,g ,arg1))
       ,@args
  ,g)))

(defmacro -prog2 (arg1 arg2 &rest args)
  (let ((g (gensym)))
```

```
        `(let ((,g (progn ,arg1 ,arg2)))
        ,@args
  ,g)))

(defmacro -progn (&rest args) `(let nil ,@args))

(defun -rem (n m)
        (nth-value 1 (truncate n m)))

(defun -stringp (x) (typep x 'string))

(defmacro -dolist ((var lst &optional result) &rest body)
   (let ((g (gensym)))
        `(do ((,g ,lst (cdr ,g)))
                 ((atom ,g) (let ((,var nil)) ,result))
                (let ((,var (car ,g)))
        ,@body))))

(defmacro -unless (arg &rest body)
        `(if (not ,arg)
                (progn ,@body)))

(defmacro -when (arg &rest body)
        `(if ,arg (progn ,@body)))
```