Programare Funcţională – Laborator 6

Închideri funcţionale. Abstractizare

Isabela Drămnesc

Martie 2023

1 Concepte

- Implementare funcţionale
- apply, filter, map, foldr, foldl, pairs
- Închideri funcţionale

2 Discutarea temei din laboratorul 5

3 Exerciţii

3.1 Implementarea funcţionalei map

```
(define (map1 f L)
    (if (null? L)
        L
        (cons (f (car L)) (map1 f (cdr L)))))

(check-expect (map1 (lambda (x) (* 2 x)) '(1 2 3 4 5)) '(2 4 6 8 10))
```

3.2 Implementarea funcţionalei filter

```
(define (filter1 p L)
    (cond ((null? L) L)
              ((p (car L)) (cons (car L) (filter1 p (cdr L))))
             (else (filter1 p (cdr L)))))

(filter1 even? '(1 2 3 4 5 6 7 8))

(filter1 odd? '(10 1 2 2 3 4))
```

3.3 Suma numerelor unei liste

Utilizând apply

```
; ------- 1) --------
(define (sumanr1 l)
   (apply + l))

(sumanr1 '(1 2 3 4 5 6))

;(sumanr1 '(1 2 a 4 7))

;(sumanr1 '(1 2 (3 4) 5 7))

; ------- 2) --------
(define (sumanr2 l)
   (cond [(list? l) (apply + l)]
              [#t error "Argumentul nu e lista!"])
)

(sumanr2 '(1 2 3 4 5))

(sumanr2 1)

; ------- 3) ----------
; argument de tip rest (definitia funcall din lab 5)

(define (sumanr3 . numbers)
     (apply + numbers))

(sumanr3 1 2 3 4 5 5)

; ------- 4) ----------
; adunam doar numerele dintr-o lista

(define (sumanr4 l)
    (cond [(cons? l) (apply + (filter (lambda (x) (number? x)) l))]
          [#t error]))

(sumanr4 '(1 2 3 4))

(sumanr4 '(1 a b 4))

(sumanr4 '())

(sumanr4 "lalala")

; ------ 5) ----------
; adaugam un filtru pentru a aduna doar numere
```

```
(define (sumanr5 . numbers)
   (apply + (filter (lambda (x) (number? x)) numbers)))

(sumanr5 1 2 3 4 'a 'b 5 5 "lalala" 45)
```

Utilizând foldr

```
; ----- 6) ----------
; utilizand foldr

(define (sumlistf L)
    (foldr + 0 L))

(sumlistf '(1 2 3 4 5))

;(sumlistf '(1 2 3 a 4 5))

; ---- 7) --------
; redefiniti functia de mai sus (adaugati un filtru pentru a aduna doar numere)
; si anume

(check-expect (sumlistf2 '(1 2 3 a 4 5 "la" 5)) 20)
```

Utilizând foldl

```
; ---- 8) --------
; cu filtru

(define (sumlistf3 L)
   (foldr + 0 (filter (lambda (x) (number? x)) L)))

(sumlistf3 '(1 2 3 4 5 10))

(sumlistf3 '(1 2 3 a 4 5 10 "la" 5))
```

3.4 Inversarea unei liste utilizând foldl

```
(define (inversaf L)
   (foldl cons null L))

(inversaf '(1 2 3 4 5 6))

; (inversaf "lalala")
```

Modificati definitia de mai sus

- verificaţi dacă argumentul e o listă,

- iar dacă nu e listă, atunci generaţi o eroare cu un mesaj “Argumentul nu e listă!”

```
(check-expect (inversaft "lalala") "Argumentul nu e lista!")
```

3.5 Definiţi în cel puţin trei moduri o funcţie care primeşte ca argument o listă şi verifică dacă lista e palindrom.

```
(check-expect (e-palindrom '(a n a)) #t)

(check-expect (e-palindrom '(l a l a l a)) #f)

(check-expect (e-palindrom "la la la") "Argumentul nu e o lista!")

; ------ 1) Folositi foldl si foldr

; ------ 2) ....

; ------ 3) ....
```

3.6 Operaţii cu numere raţionale

Reprezentăm numerele raţionale ca perechi $'(numarator\ .\ numitor)$.
Definiţi funcţii pentru:

- Extragere numărător;
- Extragere numitor;
- Afişare sub formă de fracţie;
- Transformarea din numere zecimale în numerele raţionale şi invers;
- Testare dacă două numere raţionale sunt egale;
- Adunarea, scăderea, înmulţirea şi împărţirea a două numere raţionale.

```
(check-expect (make-frac 42 62) '(21 . 31))

(check-expect (numarator '(3 . 4)) 3)

(check-expect (numitor '(3 . 4)) 4)

(check-expect (add '(6 . 5) '(3 . 7)) '(57 . 35))

(check-expect (scad '(6 . 5) '(3 . 7)) '(27 . 35))

(check-expect (inm '(13 . 7) '(14 . 39)) '(2 . 3))

(check-expect (imp '(25 . 9) '( 18 . 3)) '(25 . 54))

(check-expect (egale '(26 . 22) '( 13 . 11)) #t)
```

4 TEMA

4.1 Operaţii cu numere complexe

Reprezentăm numerele complexe ca perechi $'(real \ . \ imaginar)$
Definiţi funcţii pentru:

- Extragere parte reală;
- Extragere parte imaginară;
- Afişare sub formă de număr complex ($c = a + b * i$);
- Testare dacă două numere complexe sunt egale.
- Adunarea, scăderea, înmulţirea şi împărţirea a două numere complexe.

4.2 Perechi de elemente

Se dau două liste $'(a_1 \ a_2 \ ... \ a_n)$ şi $'(b_1 \ b_2 \ ... \ b_n)$. Definiţi funcţia care construieşte lista $((a_1 \ . \ b_1) \ (a_2 \ . \ b_2) \ ... \ (a_n \ . \ b_n))$.

4.3 Definiţi în cel puţin 5 moduri funcţii pentru:

- Calcularea sumei cifrelor unui număr;
- Calcularea numărului de cifre ale unui număr;
- Aflarea maximului dintr-o listă;
- Eliminarea maximului dintr-o listă.

Notă: Termen de realizare: laboratorul următor.