Списъци
част 3: функции от по-висок ред

доц. Атанас Семерджиев

1

Съдържание

- map
- filter
- foldl/foldr
- apply

2

2

(map op L)

```
; Прилага op върху всеки от елементите на L
(define (map op L)
   (if (null? L) '()
       (cons (op (car L))
             (map op (cdr L)) )))
```

```
(map 1+ '(1 2 3 4))  →  (2 3 4 5)
```

3

3

Вградената операция `map` може да получи N на брой списъка, с еднаква дължина.

Съответно и `op` трябва да бъде N-местна функция.

```
(map + '(1 2 3) '(10 20 30))  →  (11 22 33)
```

```
(map + '(1 2 3) '(10 20 30 40))  ✕
```

4

4

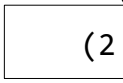
(filter pred? L)

```
; Връща списък от онези елементи на
; L, за които е верен pred?
(define (filter pred? L)
  (cond
    ((null? L) '())

    ((pred? (car L))
       (cons (car L)
             (filter pred? (cdr L))))

    (else
      (filter pred? (cdr L)))))
```

```
(filter even?
        '(1 2 3 4))
```



```
(2 4)
```

5

5

(apply op args)

Прилага `op` върху списъка от аргументи `args`.

```
(apply + '(1 2 3))
```

```
6
```

```
(apply + '())
```

```
0
```

6

6

Folds

Нека е даден списък $(e_1\ e_2\ e_3\ \dots\ e_n)$, операция op и елемент $init$.

Left fold:

$$(e_n\ op\ (e_{n-1}\ op\ (e_{n-2}\ op\ \dots\ op\ (e_1\ op\ init)\ \dots\ )))$$

```
(foldl - 0 '(1 2 3 4))  →  4-(3-(2-(1-0)))  →  2
```

Right fold:

$$(e_1\ op\ (e_2\ op\ (e_3\ op\ \dots\ op\ (e_n\ op\ init)\ \dots\ )))$$

```
(foldr - 0 '(1 2 3 4))  →  1-(2-(3-(4-0)))  →  -2
```

7

7

Left fold:

```
(foldl cons '() '(1 2 3))
        ↓
(cons 3 (cons 2 (cons 1 '())))  →  (3 2 1)
```

Right fold:

```
(foldr cons '() '(1 2 3))
        ↓
(cons 1 (cons 2 (cons 3 '())))  →  (1 2 3)
```

8

8

Списък:

```
(list 1 2 3)  →  (cons 1 (cons 2 (cons 3 '())))
```

Left fold:

```
(foldl op init '(1 2 3))  →  (op 3 (op 2 (op 1 init)))
```

Right fold:

```
(foldr op init '(1 2 3))  →  (op 1 (op 2 (op 3 init)))
```

9

9

10