Работа с матрици

доц. Атанас Семерджиев

1

Представяне на матрица

Матрица от естествени числа M ще представяме като списък от нейните редове. Например списъкът:

```
((1 2 3 4) (5 6 7 8))
```

представя матрицата:

$$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{pmatrix}$$

2

2

Намиране на размери

```
(define (number-rows M)
  (length M)
)

(define (number-columns M)
  (if (null? M) 0
      (length (car M))
  )
)
```

3

3

Извличане на редове

```
(define (first-row M)
  (car M))

(define (remove-first-row M)
  (cdr M))

(define (get-row M n)
  (list-ref M n))
```

4

4

Извличане на колони

```
(define (first-col M)
  (map car M))

(define (remove-first-col M)
  (map cdr M))

(define (get-column M n)
  (map (lambda (t) (list-ref t n)) M))
```

5

5

Извличане на елемент

```
(define (get-element M n m)
  (list-ref (list-ref M n) m)
)
```

6

6

Премахване на елемент от списък

```
(define (remove-nth l n)
  (cond
    ((null? l) l)
    ((= n 0) (cdr l))
    (else (cons (car l)
                (remove-nth (cdr l) (- n 1))))))
```

7

7

Премахване на редове и колони

```
(define (remove-row M n)
  (remove-nth M n)
)

(define (remove-col M n)
  (map (lambda (t) (remove-nth t n)) M)
)
```

8

8

Транспониране

```
(define (transpose M)
   (apply map list M)
)
```

9

9

map

`map` може да прилага функции на повече от един аргумент!

```
(map + '(1 2 3) '(4 5 6))
```

↓

```
(list (+ 1 4) (+ 5 2) (+ 3 6))
```

↓

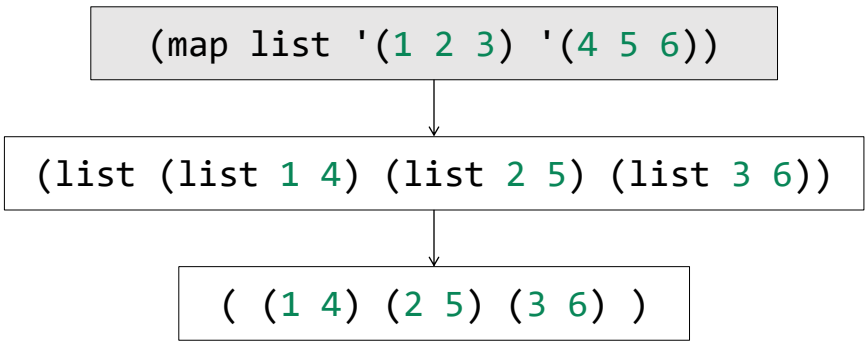
```
(5 7 9)
```

10

10

map

Респективно за `list`:

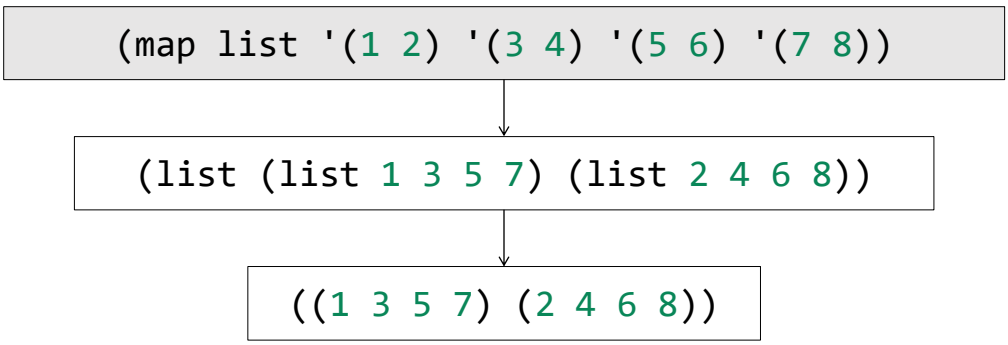


11

11

map + list

`list` може да работи върху произволен брой аргументи.



12

12

map + list

В случая само `map` и `list` не вършат работа:

```
(map list '((1 2 3) (4 5 6)) )
```

↓

```
( ((1 2 3)) ((4 5 6)) )
```

13

13

list*

Подобно на `list`, но последният аргумент се използва за опашка на списъка, а не като негов последен елемент.

`(list 1 2 '(3 4))`	`(1 2 (3 4))`
`(list* 1 2 '(3 4))`	`(1 2 3 4)`
`(list 1 2)`	`(1 2)`
`(list* 1 2)`	`(1 . 2)`

14

14

apply

Общ вид:

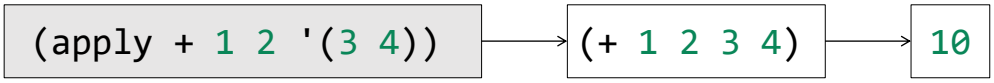
(apply op arg_1 arg_2 … arg_n args)

Прилага op върху списъка аргументи получен от:

(list* arg_1 arg_2 … arg_n args)

или все едно:

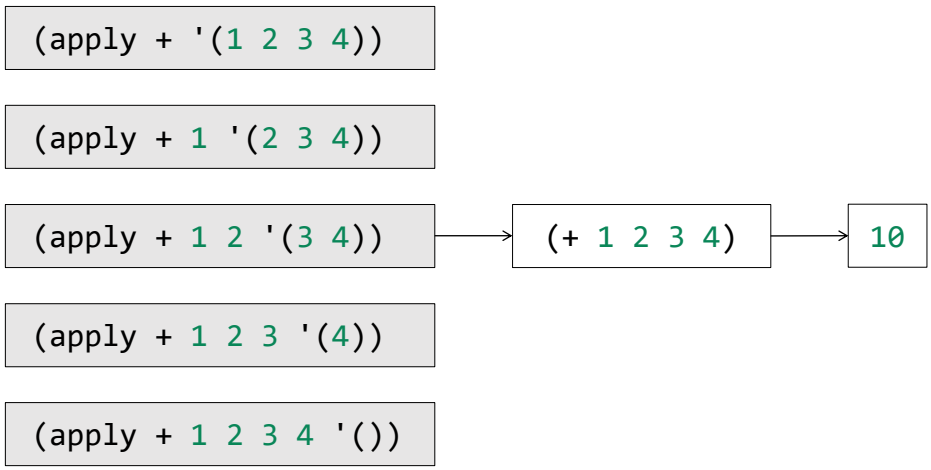
(append (list arg_1 arg_2 … arg_n) args)



15

15

apply



16

16

apply + map + list

```
(define M '( (1 2 3 4)
             (5 6 7 8) ))
```

```
(apply map list M)
```

↓

```
(map list '(1 2 3 4) '(5 6 7 8))
```

↓

```
((1 5) (2 6) (3 7) (4 8))
```

17

17

18