

Мутираци операции

Трифон Трифонов

Функционално програмиране, 2018/19 г.

14 ноември 2018 г.

Мутиращи операции в Scheme

Мутиращите операции в Scheme позволяват въвеждането на странични ефекти.

Мутиращи операции в Scheme

Мутиращите операции в Scheme позволяват въвеждането на странични ефекти.

Преглед:

- `set!` — промяна на оценка, свързана със символ

Мутиращи операции в Scheme

Мутиращите операции в Scheme позволяват въвеждането на странични ефекти.

Преглед:

- `set!` — промяна на оценка, свързана със символ
- `set-car!`, `set-cdr!` — промяна на компоненти на точкови двойки

Мутиращи операции в Scheme

Мутиращите операции в Scheme позволяват въвеждането на странични ефекти.

Преглед:

- `set!` — промяна на оценка, свързана със символ
- `set-car!`, `set-cdr!` — промяна на компоненти на точкови двойки
- `begin` — последователност от действия

Мутиращи операции в Scheme

Мутиращите операции в Scheme позволяват въвеждането на странични ефекти.

Преглед:

- `set!` — промяна на оценка, свързана със символ
- `set-car!`, `set-cdr!` — промяна на компоненти на точкови двойки
- `begin` — последователност от действия
- `open-input-file`, `open-output-file` — работа с файлове

Мутиращи операции в Scheme

Мутиращите операции в Scheme позволяват въвеждането на странични ефекти.

Преглед:

- `set!` — промяна на оценка, свързана със символ
- `set-car!`, `set-cdr!` — промяна на компоненти на точкови двойки
- `begin` — последователност от действия
- `open-input-file`, `open-output-file` — работа с файлове
- `read`, `write`, `display` — вход и изход

Промяна на оценка, свързана със символ (set!)

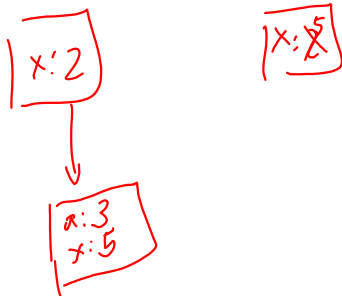
- (set! <символ> <израз>)

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>



Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) a \rightarrow 2

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) a \rightarrow 2
 - (set! a 5) a \rightarrow 5

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) a \rightarrow 2
 - (set! a 5) a \rightarrow 5
 - (define (sum x) (begin (set! a (+ a x)) a))

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) a \rightarrow 2
 - (set! a 5) a \rightarrow 5
 - (define (sum x) (set! a (+ a x)) a)

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) a \rightarrow 2
 - (set! a 5) a \rightarrow 5
 - (define (sum x) (set! a (+ a x)) a)
 - (sum 10) \rightarrow 15

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) a \rightarrow 2
 - (set! a 5) a \rightarrow 5
 - (define (sum x) (set! a (+ a x)) a)
 - (sum 10) \rightarrow 15
 - (sum 10) \rightarrow 25

Промяна на оценка, свързана със символ (set!)

- (set! <символ> <израз>)
- Търси се <символ> във веригата от среди
 - Ако бъде намерен, свързва се с оценката на <израз>
 - В противен случай — **грешка!**
- Примери:
 - (define a 2) a \rightarrow 2
 - (set! a 5) a \rightarrow 5
 - (define (sum x) (set! a (+ a x)) a)
 - (sum 10) \rightarrow 15
 - (sum 10) \rightarrow 25
 - **губи се референциалната прозрачност!**

Пример: текуща сметка

```
(define (make-account sum)
  (lambda (amount)
    (if (< (+ amount sum) 0)
        (display "Insufficient funds!\n")
        (set! sum (+ sum amount)))
    sum))
```

Пример: текуща сметка

```
(define (make-account sum)
  (lambda (amount)
    (if (< (+ amount sum) 0)
        (display "Insufficient funds!\n")
        (set! sum (+ sum amount)))
    sum))
```

- (define account (make-account 100))

Пример: текуща сметка

```
(define (make-account sum)
  (lambda (amount)
    (if (< (+ amount sum) 0)
        (display "Insufficient funds!\n")
        (set! sum (+ sum amount)))
    sum))
```

- (define account (make-account 100))
- (account 20) → 120

Пример: текуща сметка

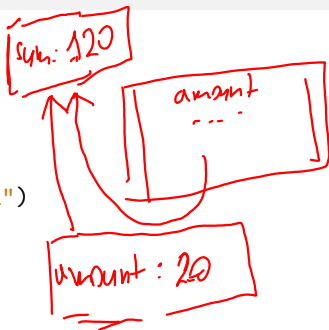
```
(define (make-account sum)
  (lambda (amount)
    (if (< (+ amount sum) 0)
        (display "Insufficient funds!\n")
        (set! sum (+ sum amount)))
    sum))
```

- `(define account (make-account 100))`
- `(account 20)` \longrightarrow 120
- `(account -50)` \longrightarrow 70

Пример: текуща сметка

```
(define (make-account sum)
  (lambda (amount)
    (if (< (+ amount sum) 0)
        (display "Insufficient funds!\n")
        (set! sum (+ sum amount))))
  sum))
```

- (define account (make-account 100))
- (account 20) → 120
- (account -50) → 70
- (account -150) → "Insufficient funds"
70



Промяна на компоненти (set-car!, set-cdr!)

- (set-car! <двойка> <израз>)
- (set-cdr! <двойка> <израз>)

Промяна на компоненти (set-car!, set-cdr!)

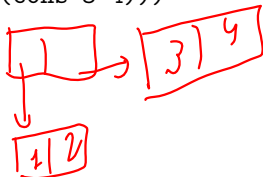
- (set-car! <двойка> <израз>)
- (set-cdr! <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>

Промяна на компоненти (set-car!, set-cdr!)

- (set-car! <двойка> <израз>)
- (set-cdr! <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**

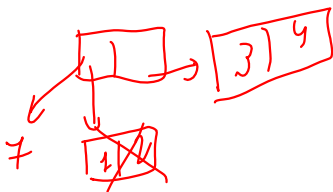
Промяна на компоненти (set-car!, set-cdr!)

- (set-car! <двойка> <израз>)
- (set-cdr! <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (define p (cons (cons 1 2) (cons 3 4)))



Промяна на компоненти (`set-car!`, `set-cdr!`)

- (`set-car!` <двойка> <израз>)
- (`set-cdr!` <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (`define` p (cons (cons 1 2) (cons 3 4)))
 - (`set-car!` p 7)



Промяна на компоненти (`set-car!`, `set-cdr!`)

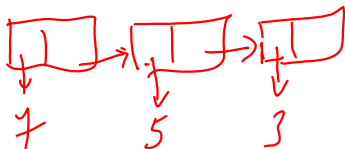
- (`set-car!` <двойка> <израз>)
- (`set-cdr!` <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (`define` p (cons (cons 1 2) (cons 3 4)))
 - (`set-car!` p 7)
 - p \rightarrow ?

Промяна на компоненти (`set-car!`, `set-cdr!`)

- (`set-car!` <двойка> <израз>)
- (`set-cdr!` <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (`define` p (cons (cons 1 2) (cons 3 4)))
 - (`set-car!` p 7)
 - p \rightarrow (7 . (3 . 4))

Промяна на компоненти (set-car!, set-cdr!)

- (set-car! <двойка> <израз>)
- (set-cdr! <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (define p (cons (cons 1 2) (cons 3 4)))
 - (set-car! p 7)
 - p \rightarrow (7 . (3 . 4))
 - (set-cdr! p '(5 3))



Промяна на компоненти (`set-car!`, `set-cdr!`)

- (`set-car!` <двойка> <израз>)
- (`set-cdr!` <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (`define` p (cons (cons 1 2) (cons 3 4)))
 - (`set-car!` p 7)
 - p \rightarrow (7 . (3 . 4))
 - (`set-cdr!` p '(5 3))
 - p \rightarrow ?

Промяна на компоненти (`set-car!`, `set-cdr!`)

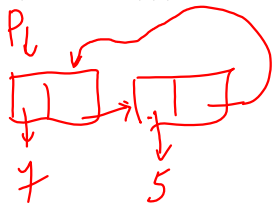
- (`set-car!` <двойка> <израз>)
- (`set-cdr!` <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (`define` p (cons (cons 1 2) (cons 3 4)))
 - (`set-car!` p 7)
 - p \rightarrow (7 . (3 . 4))
 - (`set-cdr!` p '(5 3))
 - p \rightarrow (7 5 3)

Промяна на компоненти (`set-car!`, `set-cdr!`)

- (`set-car!` <двойка> <израз>)
- (`set-cdr!` <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (`define` p (`cons` (`cons` 1 2) (`cons` 3 4)))
 - (`set-car!` p 7)
 - p \rightarrow (7 . (3 . 4))
 - (`set-cdr!` p '(5 3))
 - p \rightarrow (7 5 3)
 - (`set-cdr!` (`cdr` p) p)

Промяна на компоненти (set-car!, set-cdr!)

- (set-car! <двойка> <израз>)
- (set-cdr! <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (define p (cons (cons 1 2) (cons 3 4)))
 - (set-car! p 7)
 - $p \rightarrow (7 . (3 . 4))$
 - (set-cdr! p '(5 3))
 - $p \rightarrow (7 5 3)$
 - (set-cdr! (cdr p) p)
 - $p \rightarrow ?$



Промяна на компоненти (`set-car!`, `set-cdr!`)

- (`set-car!` <двойка> <израз>)
- (`set-cdr!` <двойка> <израз>)
- Съответният компонент на <двойка> се променя да сочи към оценката на <израз>
- **Примери:**
 - (`define` p (cons (cons 1 2) (cons 3 4)))
 - (`set-car!` p 7)
 - p \rightarrow (7 . (3 . 4))
 - (`set-cdr!` p '(5 3))
 - p \rightarrow (7 5 3)
 - (`set-cdr!` (cdr p) p)
 - p \rightarrow (7 5 7 5 7 5 ...)