

# Функции от по-висок ред

Трифон Трифонов

Функционално програмиране, 2017/18 г.

26 октомври 2017 г.

# Подаване на функции като параметри

В Scheme функциите са “първокласни” стойности.

## Примери:

- `(define (fixed-point? f x) (= (f x) x))`
- `(fixed-point? sin 0) → #t`
- `(fixed-point? exp 1) → #f`
- `(fixed-point? expt 0) → Грешка!`
- `(define (branch p? f g x) ((if (p? x) f g) x))`
- `(branch odd? exp fact 4) → 24`
- `(define (id x) x)`
- `(branch number? log id "1") → "1"`
- `(branch string? number? procedure? symbol?) → #t`

## Функции от по-висок ред

### Дефиниция

Функция, която приема функция за параметър се нарича *функция от по-висок ред*.

- fixed-point? и branch са функции от по-висок ред
- Примери за математически функции от по-висок ред?
- Всички функции в  $\lambda$ -смятането са от по-висок ред!

# Задачи за сумиране

**Задача:** Да се пресметнат следните суми:

- ❶  $k^2 + (k + 1)^2 + \dots + 100^2$  за  $k \leq 100$
- ❷  $\int_a^b f(x) \approx \Delta x [f(a) + f(a + \Delta x) + f(a + 2\Delta x) + \dots + f(b)]$
- ❸  $x + e^x + e^{e^x} + e^{e^{e^x}} + \dots$  докато поредното събираемо е  $\leq 10^{1000}$

```
(define (sum1 k)
  (if (> k 100) 0 (+ (* k k) (sum1 (+ k 1)))))
```

```
(define (sum2 a b f dx)
  (if (> a b) 0 (+ (* dx (f a)) (sum2 (+ a dx) b f dx))))
```

```
(define (sum3 x)
  (if (> x (expt 10 1000)) 0 (+ x (sum3 (exp x)))))
```

# Обобщена функция за сумиране

Да се напише функция от по-висок ред `sum`, която пресмята сумата:

$$\sum_{\substack{i=a \\ i \leftarrow \text{next}(i)}}^b \text{term}(i).$$

```
(define (sum a b term next)
  (if (> a b) 0 (+ (term a) (sum (next a) b term next))))
```

# Приложения на sum

Решение на задачите за суми чрез sum:

$$\sum_{i=k}^{100} i^2$$

```
(define (square x) (* x x))
(define (1+ x) (+ x 1))
(define (sum1 k) (sum k 100 square 1+))
```

$$\Delta x \sum_{\substack{i=a \\ i \rightarrow i+\Delta x}}^b \Delta x f(i)$$

```
(define (sum2 a b f dx)
  (define (term x) (* dx (f x)))
  (define (next x) (+ x dx))
  (* dx (sum a b term next)))
```

$$\sum_{\substack{i=x \\ i \rightarrow e^i}}^{10^{1000}} i$$

```
(define (sum3 x)
  (sum x (expt 10 1000) id exp))
```

# Обобщена функция за произведение

Да се напише функция от по-висок ред `product`, която пресмята:

$$\prod_{\substack{i=a \\ i \leftarrow \text{next}(i)}}^b \text{term}(i).$$

```
(define (prod a b term next)
  (if (> a b) 1 (* (term a) (prod (next a) b term next))))
```

```
(define (sum a b term next)
  (if (> a b) 0 (+ (term a) (sum (next a) b term next))))
```

## Обобщена функция за натрупване

Да се напише функция, която пресмята

$$term(a) \oplus \left( term(next(a)) \oplus \left( \dots \oplus (term(b) \oplus \perp) \dots \right) \right),$$

където  $\oplus$  е бинарна операция,

$a \perp$  е нейната “нулева стойност”, т.е.  $x \oplus \perp = x$ .

```
(define (accumulate op nv a b term next)
  (if (> a b) nv
      (op (term a) (accumulate op nv (next a) b term next))))

(define (sum a b term next) (accumulate + 0 a b term next))
(define (product a b term next) (accumulate * 1 a b term next))
```



## Задача: пресмятане на полином

Да се пресметне стойността на полинома

$$\begin{aligned}P_n(x) &= x^n + 2x^{n-1} + \dots + (n-2)x^3 + (n-1)x^2 + nx + (n+1) \\ &= \sum_{i=0}^n (n+1-i)x^i\end{aligned}$$

Решение №1:

```
(define (p n x)
  (define (term i) (* (- (1+ n) i) (expt x i)))
  (accumulate + 0 0 n term 1+))
```

Можем ли да решим задачата без да извикваме `expt` на всяка стъпка?

# Правило на Хорнер

$$\begin{aligned}
 P_n(x) &= x^n + 2x^{n-1} + \dots + (n-2)x^3 + (n-1)x^2 + nx + (n+1) \\
 &= \left( \left( \left( \dots ((x+2)x+3)x + \dots \right) x + (n-1) \right) x + n \right) x + (n+1)
 \end{aligned}$$

Можем ли да сметнем с `accumulate`?

Идея: Да използваме операцията  $a \oplus b := ax + b$ .

Коя е “нулевата стойност”  $\perp$ ?

Решение №2:

```

(define (p n x)
  (define (op a b) (+ (* a x) b))
  (accumulate op 0 1 (1+ n) id 1+))

```

Не смята правилно!

# Правило на Хорнер

Всъщност пресметнахме:

$$Q_n(x) = x + 2x + 3x + \dots + nx + (n+1)x = \frac{(n+1)(n+2)}{2}x.$$

Идея: Да използваме операцията  $a \oplus b := a + bx$ .

Решение №3:

```
(define (p n x)
  (define (op a b) (+ a (* b x)))
  (accumulate op 0 1 (1+ n) id 1+))
```

Пак не смята правилно!!!

## Ляво и дясно натрупване

Всъщност пресметнахме:

$$\begin{aligned} R_n(x) &= 1 + x \left( 2 + x \left( \dots + x \left( (n-1) + x(n + x(n+1)) \right) \dots \right) \right) \\ &= (n+1)x^n + nx^{n-1} + (n-1)x^{n-2} + \dots + 3x^2 + 2x + 1 \end{aligned}$$

ВМЕСТО

$$\begin{aligned} P_n(x) &= \left( \left( \left( \dots ((x+2)x+3)x + \dots \right) x + (n-1) \right) x + n \right) x + (n+1) \\ &= x^n + 2x^{n-1} + \dots + (n-2)x^3 + (n-1)x^2 + nx + (n+1). \end{aligned}$$

За неасоциативни операции  $\oplus$  има значение в какъв ред са скобите!

## Обобщена функция за ляво натрупване

Да се напише функция, която пресмята **ляво натрупване**:

$$\left( \dots \left( (\perp \oplus \text{term}(a)) \oplus \text{term}(\text{next}(a)) \right) \oplus \dots \right) \oplus \text{term}(b)$$

```
(define (accumulate-i op nv a b term next)
  (if (> a b) nv
      (accumulate-i op (op nv (term a)) (next a) b term next)))
```

- `accumulate` — дясно натрупване, рекурсивен процес
- `accumulate-i` — ляво натрупване, итеративен процес

# Правило на Хорнер

$$\begin{aligned}
 P_n(x) &= x^n + 2x^{n-1} + \dots + (n-2)x^3 + (n-1)x^2 + nx + (n+1) \\
 &= \left( \left( \left( \dots ((x+2)x+3)x + \dots \right) x + (n-1) \right) x + n \right) x + (n+1)
 \end{aligned}$$

Идея: използваме `accumulate-i` и  $a \oplus b := ax + b$ .

**Решение №4:**

```

(define (p n x)
  (define (op a b) (+ (* a x) b))
  (accumulate-i op 0 1 (1+ n) id 1+))

```

# Анонимни функции

Можем ли да ги конструираме параметрите на функциите от по-висок ред “на място”, без да им даваме имена?

- `(lambda ({<параметър>}) <тяло>)`
- Оценява се до функционален обект със съответните параметри и тяло
- Анонимната функция пази указател към средата, в която е оценена
- Примери:
  - `(lambda (x) (+ x 3)) → #<procedure>`
  - `((lambda (x) (+ x 3)) 5) → 8`
  - `(define (<име> <параметри>) <тяло>)`  
 $\iff$   
`(define <име> (lambda (<параметри>) <тяло>))`

# Примери

```
(define (integral a b f dx)
  (* dx (accumulate + 0 a b f (lambda (x) (+ x dx))))))
```

```
(define (p n x)
  (accumulate-i + 0 1 (+ n 1) (lambda (a b) (+ (* a x) b))
    (lambda (i) (+ i 1))))
```

Задача: Как можем да реализираме с `accumulate`:

- $n!$
- $x^n$
- $\sum_{i=0}^n \frac{x^i}{i!}$