Функции от по-висок ред – част 1

Трифон Трифонов

Функционално програмиране, 2025/26 г.

20 октомври 2025 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)` $\longrightarrow$ `#t`
- `(fixed-point? exp 1)` $\longrightarrow$ `#f`
- `(fixed-point? expt 0)` $\longrightarrow$ Грешка!
- `(define (branch p? f g x) ((if (p? x) f g) x))`
- `(branch odd? exp fact 4)` $\longrightarrow$ `24`
- `(define (id x) x)`
- `(branch number? log id "1")` $\longrightarrow$ `"1"`
- `(branch string? number? procedure? symbol?)` $\longrightarrow$ `#t`

Функции от по-висок ред

Дефиниция

Функция, която приема функция за параметър или връща функция като резултат 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 λ-смятането са от по-висок ред!

Задачи за сумиране

Задача: Да се пресметнат следните суми:

1. $k^2 + (k+1)^2 + \ldots + 100^2$ за $k \leq 100$
2. $\int_a^b f(x)dx \approx \Delta x\big[f(a) + f(a+\Delta x) + f(a+2\Delta x) + \ldots + f(b)\big]$
3. $x + e^x + e^{e^x} + e^{e^{e^x}} + \ldots$ 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:

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

```
(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+))
```

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

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

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

```
(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:

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

```
(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

$$term(a) \oplus \Big(term\big(next(a)\big) \oplus \Big(\ldots \oplus \big(term(b) \oplus \bot\big) \ldots \Big) \Big),$$

където $\oplus$ е двуместна операция,
а $\bot$ е нейната „нулева стойност“, т.е. $x \oplus \bot = x$.

```
(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))
```

Анонимни функции

Можем да 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))` $\longrightarrow$ `#<procedure>`
  - `((lambda (x) (+ x 3)) 5)` $\longrightarrow$ `8`
  - `(define (<име> <параметри>) <тяло>)`
    $\Longleftrightarrow$
    `(define <име> (lambda (<параметри>) <тяло>))`

Примери

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

Задача: Как можем да реализираме с accumulate:

- $n!$
- x^n
- $\sum_{i=0}^{n} \frac{x^i}{i!}$
- $\exists x \in [a; b]\ p(x)$