Лабораторная работа № 2. Рекурсия, процедуры высшего порядка, обработка списков

30 сентября 2024 г.

Булат Валиуллин, ИУ9-11Б

Цель работы

Приобретение навыков работы с основами программирования на языке Scheme: использование рекурсии, процедур высшего порядка, списков.

Индивидуальный вариант

uniq, delete, polynom, intersperse, all?, o

Реализация

```
(define (uniq xs)
  (cond
    ((= 0 (length xs)) '())
    ((= 1 (length xs)) xs)
    ((equal? (car xs) (car (cdr xs))) (uniq (cdr xs)))
    (else (cons (car xs) (uniq (cdr xs))))))


(define (delete pred? xs)
  (cond
    ((null? xs) '())
    ((pred? (car xs)) (delete pred? (cdr xs)))
    (else (cons (car xs) (delete pred? (cdr xs))))))


(define (polynom ks x)
  (if (null? ks)
      0
```

```
      (+ (* (car ks) (expt x (- (length ks) 1))) (polynom (cdr ks) x))))


(define (intersperse e xs)
  (cond
    ((null? xs) '())
    ((= 1 (length xs)) xs)
    (else (cons (car xs) (cons e (intersperse e (cdr xs)))))))


(define (all? pred? xs)
  (or (= 0 (length xs)) (and (pred? (car xs)) (all? pred? (cdr xs)))))


(define (o . funcs)
  (lambda (x)
    (if (null? funcs)
        x
        ((car funcs) ((apply o (cdr funcs)) x)))))
```

Тестирование

```
Welcome to DrRacket, version 6.10.1 [3m].
Language: R5RS; memory limit: 128 MB.
> (hello "Вася")
Привет, Вася!
> (hello "Пупкин")
Привет, Пупкин!Welcome to DrRacket, version 6.3 [3m].
Language: R5RS; memory limit: 128 MB.
> (uniq '(a a b c c c d d a b a))
(a b c d a b a)
> (uniq '(1 1 2 2 2 3 4 4 1))
(1 2 3 4 1)
> (uniq '((x 7) (x 7) (y 5) (y 5)))
((x 7) (y 5))
> (uniq '(a))
(a)
> (uniq '())
()
> (delete even? '(0 1 2 3))
(1 3)
> (delete even? '(0 2 4 6))
()
> (delete even? '(1 3 5 7))
(1 3 5 7)
```

```
> (delete even? '())
()
> (polynom '(3 5 2 8) 4)
288
> (polynom '(83 -53 74) 7)
3770
> (polynom '(4) 100)
4
> (intersperse 'x '(1 2 3 4))
(1 x 2 x 3 x 4)
> (intersperse 'x '(1 2))
(1 x 2)
> (intersperse 'x '(1))
(1)
> (intersperse 'x '())
()
> (all? odd? '(1 3 5 7))
#t
> (all? odd? '(0 1 2 3))
#f
> (all? odd? '(0 2 4 6))
#f
> (all? odd? '())
#t
> (define (f x) (+ x 2))
(define (g x) (* x 3))
(define (h x) (- x))

((o f g h) 1)
-1
> ((o f g) 1)
5
> ((o h) 1)
-1
> ((o) 1)
1
```

Вывод

После многих часов страданий, кажется, наконец я понял, как работают рекурсии, как никогда раньше. Да и хоть какое-то понимание, что такое сложность и как её считать появилось. Понял, что ещё статьи на эту тему почитать не помешает.