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

30 сентября 2024 г.

Родион Лавров, ИУ9-12Б

## Цель работы

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

## Индивидуальный вариант

1. Реализуйте процедуру (uniq xs)
2. Реализуйте процедуру (delete pred? xs)
3. Реализуйте процедуру (polynom (an … a1 a0) x)
4. Реализуйте процедуру (intersperse e xs)
5. Реализуйте предикат (all? pred? xs)
6. Реализуйте композицию функций (процедур) одного аргумента

## Реализация

```
(define (uniq xs)
  (if (null? xs)
      '()
      (if (null? (cdr xs))
          xs
          (if (equal? (car xs) (cadr xs))
              (uniq (cdr xs))
              (cons (car xs) (uniq (cdr xs)))))))

(define (delete pred? xs)
  (if (null? xs)
      '()
      (if (not (pred? (car xs)))
```

```
          (cons (car xs) (delete pred? (cdr xs)))
          (delete pred? (cdr xs)))))

(define (polynom coef x)
  (if (null? coef)
      '()
      (if (null? (cdr coef))
          (car coef)
          (+ (* (car coef) (expt x (- (length coef) 1))) (polynom (cdr coef) x)))))

(define (intersperse e xs)
  (if (null? xs)
      '()
      (if (null? (cdr xs))
          xs
          (append (cons (car xs) (cons e '())) (intersperse e (cdr xs))))))

(define (all? pred? xs)
  (or (null? xs)
      (and (pred? (car xs))
           (all? pred? (cdr xs)))))

(define (f x) (+ x 2))
(define (g x) (* x 3))
(define (h x) (- x))

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

## Тестирование

```
Welcome to DrRacket, version 8.14 [cs].
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))
> (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
> ((o f g h) 1)
-1
> ((o f g) 1)
5
> ((o h) 1)
-1
> ((o) 1)
1
```

## Вывод

Я научился писать очень классные процедуры на языке Scheme и ответил на парочку интересных вопросов. И вот ещё анекдот, чтобы скрасить будни: Я знаю одну шутку про UDP Но не факт, что она до Вас дойдёт))