% Лабораторная работа № 2. Рекурсия, процедуры высшего порядка, обработка списков % 23 Сентября 2024г. % Александр Яннаев, ИУ9-11Б

Цель работы

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

Индивидуальный вариант

1. Реализовать процедуру `(uniq xs)`, удаляющую из списка `xs` *соседние* дубликаты.
2. Реализовать процедуру `(delete pred? xs)`, которая «удаляет» из списка `xs` все элементы, удовлетворяющие предикату `pred?`.
3. Реализовать процедуру `(polynom (an … a1 a0) x)`, принимающую список коэффициентов и переменную и вычисляющую значение полинома $a_n x_n + a_1 x + … a_0$.
4. Реализовать процедуру `(intersperse e xs)`, которая возвращает список, полученный путём вставки элемента `e` между элементами списка `xs`.
5. Реализовать предикат `(all? pred? xs)`, который возвращает `#t`, если все элементы списка `xs` удовлетворяет предикату `pred?`.
6. Реализовать композицию функций (процедур) одного аргумента, для чего напишите процедуру `o`, принимающую произвольное число процедур одного аргумента и возвращающую процедуру, являющуюся композицией этих процедур.

Реализация

```
#|Процедура, удаляющая соседние дубликаты.|#
;В этой процедуре мы использовали хвостовую рекурсию
;Сложность процедуры - O(n)
(define (uniq xs)
  (if (null? xs)
      '()
      (let loop ((current (car xs))
                 (rest (cdr xs))
                 (result '()))
        (cond
          ((null? rest)
           (reverse (cons current result)))
          ((equal? current (car rest))
           (loop current (cdr rest) result))
          (else
           (loop (car rest) (cdr rest) (cons current result)))))))

#|Процедура, удаляющая из списка все элементы,
подходящие под условие.|#
;В этой процедуре мы не использовали хвостовую рекурсию
;Сложность процедуры - O(n)
(define (delete pred? xs)
  (cond
    ((null? xs) '())
```

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

#|Процедура, принимающая список коэффициентов
и переменную и вычисляющая значение полинома.|#
;В этой процедуре мы не использовали хвостовую рекурсию
;Сложность процедуры - O(n)
(define (polynom coeffs x)
  (define (helper coeffs x power)
    (if (null? coeffs)
        0
        (+ (* (car coeffs) (expt x power))
           (helper (cdr coeffs) x (+ power 1)))))
  (helper (reverse coeffs) x 0))

#|Процедура, которая возвращает список, полученный
путём вставки элемента между элементами списка.|#
;В этой процедуре мы не использовали хвостовую рекурсию
;Сложность процедуры - O(n)
(define (intersperse  e xs)
  (cond
    ((null? xs) '())
    ((null? (cdr xs)) xs)
    (else (cons (car xs)
                (cons e
                      (intersperse e (cdr xs)))))))

#|Предикат, который возвращает истинну, если все
элементы списка xs удовлетворяет предуикату.|#
;В этом предикате мы не использовали хвостовую рекурсию
;Сложность предиката - O(n)
(define (all? pred? xs)
  (or (null? xs)
      (and (pred? (car xs))
           (all? pred? (cdr xs)))))

#|Композиция функций (процедур) одного аргумента,
принимающая произвольное число процедур одного
аргумента и возвращающа процедуру, являющуюся
композицией этих процедур.|#
;В этой композиции функций мы использовали хвостовую рекурсию
;Сложность композиции функций - O(1) или O(m), где m - кол-во функций
(define (o . functions)
  (lambda (n)
    (let recur ((funcs (reverse functions))
                (res n))
      (if (null? funcs)
          res
          (recur (cdr funcs) ((car funcs) res))))))
```

Тестирование

```
Welcome to DrRacket, version 8.7 [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

Я научился:

1. Использовать:
    1. рекурсии
    2. процедуры высшего порядка
    3. списки
2. Анализировать сложность программы и наличие хвостовой рекурсии