Рекурсия

BBQTD edited this page on 4 Mar 2015 · 1 revision

Рекурсия

Рекурсия - ссылка описываемого объекта самого на себя в процессе описания.

Виды рекурсии

- Простая рекурсия. Единственный рекурсивный вызов в теле функции.
- Рекурсия второго порядка. Несколько рекурсивные вызовов в теле функции.
- Взаимная рекурсия.

```
(defun my_member (el list)
  (cond ((null lst) Nil)
        ((equal el (car lst)) T)
        (T (my-member el '(cdr lst)))))
```

Классификация

###Хвостовая рекурсия

```
(def fun (x)
  (cond (end_test1 end-value1)
        (end-test2 end-value2)
        (t  (fun reduced-x))))
```

Найти первый нечётный элемент, на всех уровнях

```
(defun first-odd (lst)
  (cond ((null lst) Nil)
        ((odd (first lst)) (first lst))
        (T (first-odd (cdr lst)))))

(defun first (lst)
  (cond ((atom lst) lst)
         (T (first (car lst)))))
```

Дополняемая рекурсия

```
(defun fun (x)
  (cond (test end-value)
        (t (add-fun add-value
                    (fun changed-x)))))
```

Подсчёт `cons` -ячеек

```
(defun my-length (lst)
  (cond ((null lst) 0)
        (t (+ 1
              (my-length (cdr lst))))))
```

Сортировка методом вставки (`cons` *дополняемая рекурсия*)

```
(defun insert-help (x lst)
  (cond ((null lst) (list x))
        ((<= x (car lst)) (cons x lst))
        (T (cons (car lst)
                 (insert-help x
                              (cdr lst))))))
```

▾ Pages 3

Home

Абстрактный подход

Рекурсия

```
(defun sort-help (lst1 lst2)
  (cond ((null lst1) lst2)
        (T (sort-help (cdr lst1)
                      (insert-help (car lst1)
                                   lst2)))))

(defun sort-ins (lst)
  (sort-help lst ()))
```

Множественная рекурсия

На одной ветке сразу несколько рекурсивные вызовов (`car` - `cdr` рекурсивная обработка)

```
(defun fun (x)
  (cond (test end-value)
        (t (combiner (fun changed1-x)
                     (dun changed2-x)))))
```

Найти первое число смешанного списка

```
(defun first-number (lst)
  (cond ((number lst) lst)
        ((atom lst) nil)
        (T (or (first-number (car lst))
               (first-number (cdr lst))))))

(first-number '((a b) (c 5) t 5)) => 5


(defun cons-sells (lst)
  (if (atom lst)
      0
      (+ (length lst)
         (reduce #'+ (mapcar #'cons-sells lst)))))


(defun intaone (lst rst)
  (cond ((null lst) rst)
        ((atom lst) (cons lst rst))
        (T (abofadfadf (car lst)
                       (intaone (cdr lst)
                                rst)))))

(defun dsafadsgfa (lst)
  (intaone lst ()))
```

Итерационный циклы

```
(dotimes (var-name end-integer [result-form])
         body)


(dolist (var-name list_reduce [result-form])
        body)


(do ((var1 int1 [dasdasd])
     (var2 int2 [fadsfas])
     ...
     (varN intN [dsafasd]))
    (test addition ... additionS)
    body)
```