Практика рекурсивного программирования

Лектор: Артамонов Юрий Николаевич

Университет "Дубна"
филиал Котельники

Задача 1

Реализовать функцию $my\_member(a, L)$, которая для списка L и заданного элемента a проверяет принадлежность его списку:

$$my\_member(5, [1, 2, 3, 4, 5, 6, 7, 6, 5]) \Rightarrow T$$
$$my\_member(9, [1, 2, 3, 4, 5, 6, 7, 6, 5]) \Rightarrow NIL$$

Решение задачи 1

```
(defun my_member (a L)
  (cond
    ((Null L) nil)
    ((equal a (car l)) T)
    (t (my_member a (cdr L)))))
```

Задача 2

Реализовать функцию $my\_last(L)$, которая для списка L возвращает последний элемент списка.

$$my\_last([1, 2, 3, 4, 5, 6, 7]) \Rightarrow 7$$

Решение задачи 2

```
(defun my_last (L)
  (if (Null (cdr L))
      (car L)
      (my_last (cdr L))))
```

Задача 3

Реализовать функцию $my\_reverse(L)$, которая реверсирует список L:

$$my\_reverse([1, 2, 3, 4]) \Rightarrow [4, 3, 2, 1]$$

Решение задачи 3

```
(defun my_reverse (L)
  (cond
    ((Null L) L)
    (t (append (my_reverse (cdr L)) (list (car L))))))
```

Задача 4

Реализовать функцию $my\_last\_m(L)$ с использованием функции $reverse$.

Решение задачи 4

```
(defun my_last_m (L)
  (car (reverse L)))
```

Задача 5

Реализовать функцию $my\_nth(n, L)$, которая возвращает n-й элемент списка L:

Решение задачи 5

```
(defun my_nth (n L)
  (if (= n 0)
      (car L)
      (my_nth (- n 1) (cdr L))))
```

Задача 6

Реализовать функцию $del\_last(L)$, которая возвращает список L без последнего элемента:

Решение задачи 6

```
(defun del_last (L)
  (cond
    ((Null (cdr L)) nil)
    (t (cons (car L) (del_last (cdr L))))))
```

Задача 7

Реализовать функцию $del\_fix(n, L)$, которая возвращает список L без n-го элемента:

Решение задачи 7

```
(defun del_fix (n L)
  (cond
    ((= n 0) (cdr L))
    (t (cons (car L) (del_fix (- n 1) (cdr L))))))
```

Задача 8

Реализовать функцию $each\_even(L)$, которая возвращает список из всех элементов списка L с четными номерами:

$$each\_even([a, b, c, d, e, f]) \Rightarrow [b, d, f]$$

Решение задачи 8

```
(defun each_even (L)
  (cond
    ((Null L) nil)
    ((Null (cdr L)) (cadr L))
    (t (cons (cadr L) (each_even (cddr L))))))
```

Задача 9

Реализовать функцию $max\_elem(L)$, которая находит максимальный элемент списка L

Решение задачи 9

```
(defun max_elem (L)
  (cond
    ((Null (cdr L)) (car L))
    ((Null L) nil)
    ((>= (car L) (max_elem (cdr L))) (car L))
    (t (max_elem (cdr L)))))
```

Задача 10

Реализовать функцию $create\_set(L)$, которая удаляет из списка L все повторяющиеся элементы (оставляя каждый элемент в одном экземпляре).

Решение задачи 10

```
(defun create_set (L)
  (cond
    ((Null L) nil)
    ((member (car L) (cdr L)) (create_set (cdr L)))
    (t  (cons (car L) (create_set (cdr L))))))
```

Задача 11

Реализовать функцию $union\_set(A, B)$ объединения двух множеств A и B.

$$union\_set([1, 2, 3], [2, 3, 4]) \Rightarrow [1, 2, 3, 4]$$

Решение задачи 11

```
(defun union_set (A B)
  (create_set (append A B)))
```

Задача 12

Реализовать функцию $cross\_set(A, B)$ пересечения двух множеств A и B.

$$union\_set([1, 2, 3], [2, 3, 4]) \Rightarrow [2, 3]$$

Решение задачи 12

```
(defun cross_set (A B)
  (cond
    ((Null A) A)
    ((member (car A) B)
     (cons (car A) (cross_set (cdr A) B)))
    (t (cross_set (cdr A) B))))
```

Задача 13

Реализовать функцию $sub\_set(A, B)$ разности двух множеств A и B.

$$union\_set([1, 2, 3, 5], [2, 3, 4]) \Rightarrow [1, 5]$$

Решение задачи 13

```
(defun sub_set (A B)
  (cond
    ((Null A) A)
    ((not (member (car A) B))
     (cons (car A) (sub_set (cdr A) B)))
    (t (sub_set (cdr A) B))))
```

Задача 14

Реализовать функцию $sym\_sub\_set(A, B)$ симметрической разности двух множеств A и B.

$$sym\_sub\_set([1, 2, 3, 5], [2, 3, 4]) \Rightarrow [1, 5] \cup [4] = [1, 4, 5]$$

Решение задачи 14

```
(defun sym_sub_set (A B)
  (union_set (sub_set A B) (sub_set B A)))
```

Задача 15

Реализовать функцию $dekart\_set(A, B)$ декартова произведения множеств A и B.

$$dekart\_set([1, 2, 3], [2, 3]) \Rightarrow [(1, 2), (1, 3), (2, 2), (2, 3), (3, 2), (3, 3)]$$

Решение задачи 15

```
(defun dekart_set (A B)
  (defun iter (aa B)
    (cond
      ((Null B) nil)
      (t (cons (list aa (car B))
      (iter aa (cdr B))))))
  (cond
    ((Null A) nil)
    (t (append (iter (car A) B)
    (dekart_set (cdr A) B)))))
```

Задача 16

Реализовать функцию $make\_list(a, b)$ создания списка $[a, (a+1), (a+2), \ldots, b]$

$$make\_list(1, 5) \Rightarrow [1, 2, 3, 4, 5]$$

Решение задачи 16

```
(defun make_list (a b)
  (if (> a b)
      nil
      (cons a (make_list (+ a 1) b))))
```

Задача 17

Реализовать функцию $filter\_list(predicate, L)$, которая просеивает список L, оставляя только элементы списка, удовлетворяющие предикату $predicate$. Для примера реализовать предикаты проверки, что число является квадратом $square\_p$, или простым числом $prime\_p$

Решение задачи 17

```
(defun filter_list (predicate L)
  (cond
    ((Null L) L)
    ((eval `(,predicate (car (quote ,L))))
     (cons (car L) (filter_list predicate (cdr L))))
    (t (filter_list predicate (cdr L)))))

(defun square_p (n)
  (if ( = (expt (round (sqrt n)) 2) n) t nil))
(defun prime_p (n)
  (defun iter (a n)
    (cond
      ((= a 1) T)
      (( = (mod n a) 0) nil)
      (t (iter (- a 1) n))))
  (iter (round (sqrt n)) n))
```

Задача 18

Реализовать функцию $map\_list(func, L)$, которая применяет функцию $func$ к каждому элементу списка L и возвращает список из результатов:

$$map\_list(sqrt, [1, 2, 3]) \Rightarrow [1, 1.4142135, 1.7320508]$$

Решение задачи 18

```
(defun map_list (func L)
  (cond
    ((Null L) L)
      (t (cons (eval `(,func (car (quote ,L))))
      (map_list func (cdr L))))))

(map_list 'sqrt '(1 2 3 4 5 6))
(1.0 1.4142135 1.7320508 2.0 2.236068 2.4494898)

(map_list 'list '(1 2 3 4 5 6))
((1) (2) (3) (4) (5) (6))

 (map_list 'atom '(1 2 3 4 5 6))
 (T T T T T T)
```

Задача 19

Реализовать функцию $accumulate\_list(operation, initial, L)$, которая применяет операцию $operation$ с начального значения $initial$ над каждым элементом списка L:

$$accumulate\_list(*, 1, [1, 2, 3, 4]) \Rightarrow 1 * 1 * 2 * 3 * 4 = 24$$
$$accumulate\_list(+, 0, [1, 2, 3, 4]) \Rightarrow 0 + 1 + 2 + 3 + 4 = 10$$

Решение задачи 19

```
(defun accumulate_list (operation initial L)
  (cond
    ((Null L) initial)
    (t (eval `(,operation (car (quote ,L))
    ,(accumulate operation
    initial (cdr L)))))))
```

Задача 20

Реализовать быструю сортировку (сортировку Хоара).

Решение задачи 20

```
(defun gen_random (N M)
  (if (> N 0)
      (cons (random M) (gen_random (- N 1) M))
      nil))
(defun DevisionS1 (a L)
  (cond
    ((Null L) nil)
    ((> a (car L)) (cons (car L)
    (DevisionS1 a (cdr L))))
    (t (DevisionS1 a (cdr L)))))
(defun DevisionS2 (a L)
  (cond
    ((Null L) nil)
    ((<= a (car L)) (cons (car L)
    (DevisionS2 a (cdr L))))
      (t (DevisionS2 a (cdr L)))))
```

Решение задачи 20 (продолжение)

```
(defun SortShort (L)
  (cond
    ((Null L) nil)
    (t (append (SortShort
    (DevisionS1 (car L) (cdr L)))
    (list (car L))
    (SortShort
      (DevisionS2 (car L) (cdr L)))))))
```

Решение задачи из лекции

```
(defun Number_To_HV (p q)
       (cond
       ((and (= p 2) (= q 1))(list 'H))
       ((= p q ) nil)
       ((< p q) (cons 'V (Number_To_HV p (- q p))))
       ((> p q) (cons 'H (Number_To_HV (- p q) q)))))
```

Решение задачи из лекции (продолжение)

```
(defun HV_To_Number (L)
  (defun iter (result LL)
    (cond
      ((Null LL) result)
      ((eq (car LL) 'H)
       (iter (+ result 1) (cdr LL)))
      ((eq (car LL) 'V)
       (iter (/ result (+ result 1)) (cdr LL)))))
  (iter 1 (reverse L)))
```