Московский государственный технический университет
имени Н. Э. Баумана

Факультет: Информатика и системы управления
Кафедра: Программное обеспечение ЭВМ и информационные технологии
Дисциплина: Функциональное и логическое программирование

Лабораторная работа №7

Выполнили: Никичкин А. С., Фокеев А. И.
Группа: ИУ7–61

Москва, 2015 г.

1 Итеративный вариант функции `memberp`, которая возвращает `T`, или `NIL` в зависимости от того, принадлежит ли первый аргумент второму, как элемент

```
1 (defun memberp (el l)
2   (dolist (i l)
3     (when (equal i el)
4       (return t))))
```

2 Итеративный вариант функции `assoc`

```
1 (defun it-assoc (key table)
2   (dolist (entry table)
3     (when (equal key (first entry))
4       (return entry))))
```

3 Итеративный вариант функции `length`

```
1 (defun it-length (l)
2   (let ((n 0))
3     (dolist (i l n)
4       (incf n))))
```

4 Итеративный вариант функции `nth`

```
1 (defun it-nth (n l)
2   (declare (fixnum n))
3   (if (>= n 0)
4       (do ((x l (rest x))
5            (i 0 (1+ i)))
6           ((= i n) (first x)))))
```

5 Итеративный вариант функции `reverse`

```
1 (defun it-reverse (l)
2   (let ((result nil))
3     (dolist (item l result)
4       (push item result))))
```

6 Итеративные варианты функций, вычисляющие объединение, разность и симметрическую разность двух множеств

```lisp
(defun it-union (x y)
  (let ((result (copy-list x)))
    (dolist (item y result)
      (if (not (member item result))
        (push item result)))))

(defun it-complement (x y)
  (let ((result nil))
    (dolist (item x result)
      (when (not (member item y))
        (push item result)))))

(defun it-symmetric-difference (x y)
  (it-union (it-complement x y) (it-complement y x)))
```

7 Функция возвращающая наибольший элемент из списка чисел

7.1 с помощью `dolist`

```lisp
(defun dolist-max (l)
  (let ((result (first l)))
    (dolist (item l result)
      (when (> item result)
        (setf result item)))))

(defun do-max (l)
  (do* ((result (first l))
        (sub-l (rest l) (rest sub-l))
        (item (first sub-l) (first sub-l)))
       ((null sub-l) result)
    (when (> item result)
      (setf result item))))
```

7.2 с помощью `do`

```lisp
(defun do-max (l)
  (do* ((result (first l))
        (sub-l (rest l) (rest sub-l))
        (item (first sub-l) (first sub-l)))
       ((null sub-l) result)
    (when (> item result)
      (setf result item))))
```

8 Функция возвращающая первый нечисловой элемент из списка

8.1 с помощью `dolist`

```lisp
(defun dolist-first-not-number (l)
  (dolist (item l)
    (when (not (numberp item))
      (return item))))
```

8.2 с помощью do

```lisp
(defun do-first-not-number (l)
  (do* ((sub-l l (rest sub-l))
        (item (first sub-l) (first sub-l)))
       ((or (null sub-l) (not (numberp item))) item)))
```

8.3 с помощью рекурсии

```lisp
(defun recursive-first-not-number (l)
  (let ((item (first l)))
    (cond
      ((null l) nil)
      ((not (numberp item)) item)
      (t (recursive-first-not-number (rest l))))))
```

9 Функция сортирующая список из чисел по возрастанию

9.1 итеративный способ

```lisp
(defun bubble-sort-list (l &optional (predicate #'<))
  (do ((swapped t)) ((not swapped) l)
    (setf swapped nil)
    (do ((l l (rest l))) ((endp (rest l)))
      (when (funcall predicate (second l) (first l))
        (rotatef (first l) (second l))
        (setf swapped t)))))
```

9.2 рекурсивный способ

```lisp
(defun quicksort (l)
  (let ((pivot (first l)))
    (if (rest l)
        (nconc (quicksort (remove-if-not #'(lambda (x) (< x pivot))
                                         l))
               (remove-if-not #'(lambda (x) (= x pivot))
                              l)
               (quicksort (remove-if-not #'(lambda (x) (> x pivot))
                                         l)))
        l)))
```