Отчет по лабораторной работе № 2
по курсу «Функциональное программирование»

Студент группы 8о-306Б МАИ *Макаров Никита*, №11 по списку
Контакты: quizbeat@gmail.com
Работа выполнена: 03.04.2015

Преподаватель: Иванов Дмитрий Анатольевич, доц. каф. 806
Отчет сдан:
Итоговая оценка:
Подпись преподавателя:

1. Тема работы

Простейшие функции работы со списками Common Lisp.

2. Цель работы

Научиться конструировать списки, находить элемент в списке, использовать схему линейной и древовидной рекурсии для обхода и реконструкции плоских списков и деревьев.

3. Задание

Запрограммируйте функцию-предикат `tree-similar-p (x y)`, которая принимает два аргумента - дерева, представленных в виде списков атомов. Предикат должен вернуть истину, если одинаковые атомы расположены в списках `x` и `y` в одном и том же порядке при обходе дерева слева направо, т.е. независимо от внутренней структуры `x` и `y`.

```
(tree-similar-p '(1 (2 (3 4)) 5)) '((1 2) 3 (4 5))) => T
```

4. Оборудование студента

Процессор Intel Core i5 2 @ 1.3GHz, память: 4Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Mac OS X 10.9, среда Clozure CL 1.10

6. Идея, метод, алгоритм

Так как в задании не нужно учитывать структуру дерева, то можно заметить, что достаточно проверить на эквивалентность списки, представляющие деревья. Но

так как списки могут быть вложенными, сначала необходимо избавиться от вложенности. Функция `to-list` раскрывает все вложенные списки, а функция `equal` проверяет эквивалентность двух списков.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun to-list (tree)
    (if (null tree)
        nil
        (if (atom (first tree))
            (cons (first tree) (to-list (rest tree)))
            (append (to-list (first tree)) (to-list (rest
   tree)))
        )
    )
)

(defun tree-similar-p (tree1 tree2)
    (equal (to-list tree1) (to-list tree2))
)
```

8.2. Результаты работы

```
> (tree-similar-p '(1 (2 (3 4)) 5) '((1 2) 3 (4 5)))
T
> (tree-similar-p '(1 (2 (3 4)) 5) '((1 2) 3 (5 4)))
NIL
> (tree-similar-p '((1 2)) '(1 2))
T
> (tree-similar-p '((1) (2)) '((1) 2))
T
> (tree-similar-p '(1 (2 (3))) '(((3) 2 ) 1))
NIL
```

9. Дневник отладки

10. Замечания, выводы

Выполняя эту лабораторную работу, я познакомился с новыми для меня возможностями языка Common Lisp, получил навыки работы со списками. Списки в Lisp'е довольно мощное средство, и в то же время просты в изучении. С их помощью можно удобно представлять деревья, очень важную структуру данных.