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

Студент группы 8О-308 Петросян Виктор, № по списку 15.

Контакты: viko20000@mail.ru
Работа выполнена: 28.04.2020
Преподаватель: Иванов Дмитрий Анатольевич, доц. каф. 806
Отчет сдан:
Итоговая оценка:
Подпись преподавателя:

1. Тема работы

Простейшие функции работы со списками Коммон Лисп.

2. Цель работы

Научиться работать со списками, изучить способы навигации по списку, а также изменения значений в нём.

3. Задание (вариант № 2.26)

Запрограммировать рекурсивно на языке Коммон Лисп функцию square-tree, которая применяется к дереву чисел, представленному в виде списка с подсписками, и возвращает дерево, по структуре совпадающее с исходным, но в котором все листья возведены в квадрат.

4. Оборудование студента

Ноутбук Samsung NP300E5c-S0URU 15.6, процессор Intel® Core™ i5-2410M CPU 2.30GHz, память 6ГБ, 64-разрядная система.

5. Программное обеспечение

Linux 4.15.0-96-generic #97-Ubuntu SMP, использовал sbcl(для запуска .lisp файлов) + VSCode(для редактирования кода)

6. Идея, метод, алгоритм

Проверяю является ли объект подсписком нашего исходного списка. Если объект является подсписком, то перемещаюсь внутрь подсписка рекурсивно вызывая для него функцию square-tree, если объект не подсписок возвожу значение в квадрат.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

Программа

```
(defun makeSquare (x) (* x x))

(defun square-tree-recursive (seq)
        (if (not (null seq))
                (if (listp seq)
                        (progn
                                (setf (first seq) (square-tree-recursive (first seq)))
                                (square-tree-recursive (rest seq))
                                (return-from square-tree-recursive seq))
                        (progn
                                (setf seq (makeSquare seq))
                                (return-from square-tree-recursive seq)))))
(defun square-tree (seq)
        (setq answer (copy-tree seq))
        (return-from square-tree (square-tree-recursive answer)))
(setq tree '(1 (2 (3 4) 5) (6 7)))
(print tree)
(print (square-tree tree))
(print tree)
```

Результаты

```
(1 (2 (3 4) 5) (6 7))
(1 (4 (9 16) 25) (36 49))
(1 (2 (3 4) 5) (6 7))
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание
1	28.04 12:03	Для вложенных списков функция возвращала Nil вместо возведения соотвествующих элементов в квадрат	В 9-ю строку кода добавил (return-from square-tree seq)	Эту ошибку было не просто найти. Использовал метод отладки путём печати значений некоторых переменных.

2	30.04 23:03	Недостаток программы - она портит исходное дерево.	Написал дополнительную функцию, которая перед вызовом основной делает копию дерева и дальше работает с копией, не затирая исходные данные.	О затирании данных прежней версией программы я знал. Осознанно написал её таким образом в целях экономии памяти, чтобы всё происходило “in place”. Новая версия программы требует O(n) дополнительной памяти, но при этом не затирает исходные данные.

10. Замечания автора по существу работы

11. Выводы

В данной лабораторной работе я познакомился со списками, изучил простейшие способы объявления списка, а также навигации и изменения значения в нём. Написал программу, которая принимает в качестве параметра дерево в виде списка, а возвращает дерево, по структуре совпадающее с исходным, но в котором все листья возведены в квадрат. Программа работает правильно и прошла все тесты.