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

Студент группы М8О-307Б-18 МАИ *Скворцов Кирилл Алексеевич*, №20
Контакты: kilyla2@yandex.ru
Работа выполнена: 15.04.2021

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

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 (вариант №2.41)

Запрограммируйте рекурсивно на языке Common Lisp 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.

4. Оборудование студента

Ноутбук Xiaomi mi Pro 15.6, процессор Intel Core i7-8550U CPU 1.80GHz, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Windows 10, онлайн компилятор для common-lisp, текстовый редактор VSCode (использовал т.к. там есть встроенный синтаксический валидатор).

6. Идея, метод, алгоритм

Идея заключается в рекурсивнном обходе каждого из деревьев в порядке preorder (слева направо) и их лениаризация в список. Далее просто происходит сравнение 2х списков. Если списки равны, то и элементы деревьев при preorder обходе равны. Иначе - их порядок не одинаков.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun get_preorder_list (tree)
    (cond ((null tree) nil)
          ((atom (first tree))
          (cons (first tree) (get_preorder_list (rest tree))))
          (t (append (get_preorder_list(first tree))
   (get_preorder_list (rest tree))))))

(defun tree-similar-p (first_tree second_tree)
    (equal (get_preorder_list first_tree) (get_preorder_list
   second_tree))
    )
```

8.2. Результаты работы

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

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание
18.04.2021	Неоправданное использование глобальных переменных	Изменена функция получения списка вершин дерева	

10. Замечания автора по существу работы

Была исправлена функция получения списков вершин. В изначальном варианте она больше походила на программу, написанную на объектно-ориентированном языке программирования (в функцию передавалось несколько параметров и использовались глобальные переменные, без которых можно было легко обойтись). После исправления код стал компактнее и логичнее.

11. Выводы

Благодаря данной лабораторной работе я нучился писать привычный рекурсивные обходы деревьев на функциональном языке программирования. Было достаточно непривычно, однако заметил удобство и простоту реализации таких обходов. Помимо этого заметил удобство работы со списками, т.к. большая часть требуемых функций уже реализована и имеет интуитивно понятные названия и способы вызовов. Сложность работы программы линейна, т.к. мы посещаем каждую вершину лишь 1 раз.