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

Студент группы М8О-307Б-19 МАИ *Ежов Никита Павлович*, №9 по списку
Контакты: nikita.ejov2012@yandex.ru
Работа выполнена: 12.04.2022

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

1. Тема работы

Простейшии 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 (вариант № 2.15)

Запрограммируйте рекурсивно на языке Коммон Лисп функцию, подсчитывающую число вхождений заданного действительного числа в дерево. "Действительное"означает, что типы чисел могут смешиваться: целые, рациональные и с плавающей точкой.

4. Оборудование студента

Процессор Intel i7-4770 (8) @ 3.9GHz, память: 16 Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Kubuntu 20.04.4 LTS, комилятор GNU CLISP 2.49.92, текстовый редактор Atom 1.58.0

6. Идея, метод, алгоритм

Реализую алгоритм обхода дерева при помощи древовидной рекурсии, и каждый лист буду сравнивать с заданным в аргументе функции числом. При обходе вглубь буду также передавать число, с которым нужно сравнить лист. Если описывать написанную программу, то у нас существует функция, которая возвращает 0 в случае если данный узел не является листом или если значения листа не равно заданному аргументу и 1 в случае, если значение листа равно заданному аргументу. В качестве ответа получается сумма всех результатов вызова функции.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
;2.15

(defun count-real (num x)
  (cond ((null x) 0)
        ((atom x) (if (= x num) 1 0))
        (t (+ (count-real num (first x))
              (count-real num (rest x))))))
```

8.2. Результаты работы

```
;; Загружается файл lab2.lisp ...
;; Загружен файл lab2.lisp
[1]> (count-real 3.0 '((1 2.2) 3.0 (4 5/6 (3 6)) 7))
2
[2]> (count-real 3.0 '((1 2.2) 3.0 (4 5/6 (3 6)) 7 (3.0 5)))
3
[3]> (count-real 3.0 '((1 2.2) 3.0 (4 5/6 (3 6)) 7 (3.0 3)))
4
[4]> (count-real 3.0 '((1 2) 3 3))
2
[5]> (count-real 4.0 '(1 4 (3.0 4.0) 5/6))
2
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание
18.04.2022	Написанная программа работает некорректно, если ей передавать в качестве аргумента любое число кроме 3	Был обнаружен код, использовавшийся для отладки на конкретном примере и замёнен на такой, который работает с переданным аргументом, а не с числом 3	Ошибка была обнаружена только после замечания преподавателя

10. Замечания автора по существу работы

Из-за древовидной рекурсии возможно быстрое увелечение потребляемой памяти при сильной "ветвистости"анализируемого дерева.

11. Выводы

Я познакомился со списками и деревьями в Common Lisp. Работа с ними очень похожа на работу с аналогичными структурами в языке Prolog, поэтому работу было выполнить несложно.