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

Студент группы 8О-306 МАИ *Наумов Дмитрий*, №15 по списку
Контакты: dandachok@gmail.com
Работа выполнена: 18.04.2020

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

1. Тема работы

Простейшие функции для работы со списками Коммон Лисп

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 №38)

Запрограммируйте рекурсивно на языке Коммон Лисп функцию индивидуумы (l), которая принимает в качестве аргумента список атомов l и выдаёт в результате список всех атомов, которые встречаются в l ровно один раз.

4. Оборудование студента

Процессор Intel Core i3-2100 4 @ 2.2GHz, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 18.04.4 LTS, clisp 2.49.60

6. Идея, метод, алгоритм

Весь алгоритм реализован в функции `f`, функция `индивидуумы` лишь обертка над ней. Идея состоит в том, чтобы пройти по списку, и для каждого элемента проверить, есть ли его копия в оставшейся части списка. Но прежде чем добавлять елемент в результирующий список, так же надо узнать втречался ли он раньше. Для этого используется вспомогательный список `dublicateList`. Туда помещаются все дубликаты которые встретились нам на пути.

7. Распечатка программы и её результаты

7.1. Исходный код

```
(defun f (inputList dublicateList)
    (let ((head (first inputList)) (tail (rest inputList)))
    (if (null inputList)
        (list)
        (if (member head dublicateList)
            (f tail dublicateList)
            (if (member head tail)
                (f tail (append dublicateList (list head)))
                (append (list head) (f tail dublicateList)))))))

(defun индивидуумы (inputList)
    (f inputList (list)))
```

7.2. Результаты работы

```
dandachok@dpc:~/Documents/Study/sem6/FP/lab2$ clisp
[1]> (load "solution.lisp")
[2]> ( индивидуумы '(a b c a d c e))
(B D E)
[3]> ( индивидуумы '(a d c e))
(A D C E)
[4]> ( индивидуумы '(a))
(A)
[5]> ( индивидуумы '(a a))
NIL
[6]> ( индивидуумы '(a a b))
(B)
[7]> ( индивидуумы '(a b a b))
NIL
[12]> (exit)
```

8. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

9. Замечания автора по существу работы

Реализация списков в Лиспе, похожа на их реализацию в языке Prolog, который мы изучали на втором курсе, поэтому сложностей с работой с ними не возникло.

10. Выводы

Программа работает за квадратичное время и имеет линейную сложность по памяти. Не смотря на то, что явно функция проходит по каждому элементу списка всего раз, встроенная функция `member`, которая вызывается для каждого элемента списка, работает также за линейное время, так и получается квадратичная сложность по времени. Учитывая что в лиспе при вызове функции параметры передаются по значения и полностью копируются, сложность по памяти тоже может возрасти до квадратичной, но теоретически мы имеем всего один вспомогательный список.