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

Студент группы 8О-307 МАИ *Вельтман Лина*, №7 по списку
Контакты: kluuo@mail.ru
Работа выполнена: 27.03.2020

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

3. Задание (вариант №12)

Запрограммируйте рекурсивно на языке Коммон Лисп функцию, подсчитывающую число вхождений заданного целого числа в дерево.

4. Оборудование ПЭВМ студента

Ноутбук MacBook Pro (13-inch, 2017), процессор 2.3GHz Intel Core i5, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение ЭВМ студента

macOS Catalina 10.15.4, реализация языка SBCL 1.4.16, текстовый редактор Sublime Text 3.

6. Идея, метод, алгоритм

Функция `count-int` принимает в качестве первого аргумента целое число и в качестве второго аргумента список. Нужно найти количество вхождений первого числа в дерево. Работает она следующим образом:

1. При помощи ключевого слова `&aux` определяем локальные переменные: `head` и `tail`, которые равны первому элементу списка (атом) и оставшемуся списку без первого элемента соответственно.
2. Для реализации ветвления алгоритма я использую `cond`. В качестве первого условия идет проверка на пустоту головы списка, ведь если головы списка нет, то и самого списка тоже. Если условие выполняется, то выводим 0 на экран и заканчиваем работу.
3. Вторым условием является проверка головы на атомарность, если голова списка - атом, то я сравниваю его с заданным целым числом `number`. Если равенство выполняется, то делаю сложение единицы с рекурсивным вызовом функции `count-int` с аргументами: заданное целое число `number` и хвост списка `tail`, иначе выполняется вызов функции `count-int` с такими же аргументами, но без сложения с единицей.
4. Когда ни одно из вышеописанных условий не выполняется, то в качестве последнего условия совершается сложение двух рекурсивных вызовов функций `count-int`, аргументами первой выступают: заданное целое число `number` и голова списка `head`, второй: заданное целое число `number` и хвост списка `tail`.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
Запрограммируйте
;;; рекурсивно на языке Коммон Лисп функцию,подсчитывающую
;;; число вхождений заданного целого числа в дерево. Примеры
;;;
;;;(count-int 3 '((1 2) 3 (4 5 (3 6)) 7)) => 2


(defun count-int (number tree &aux (head (car tree)) (tail (cdr
   tree)))
  (cond ((null tree) 0)
        ((atom head) (if (eql number head)
                         (1+ (count-int number tail))
                         (count-int number tail)))
        ((+ (count-int number head) (count-int number tail)))))
```

8.2. Результаты работы

```
(base) MacBook-Pro-Lina:lab2 linuxoid$ sbcl
```

```
This is SBCL 1.4.16, an implementation of ANSI Common Lisp.
More information about SBCL is available at
   <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no
   warranty.
It is mostly in the public domain; some portions are provided
   under
BSD-style licenses.  See the CREDITS and COPYING files in the
distribution for more information.
* (compile-file "./lab2_12.lisp")
; compiling file
   "/Users/linuxoid/Desktop/VUZICH/FP/lab2/lab2_12.lisp" (written
   27 MAR 2020 03:11:10 PM):
; compiling (DEFUN COUNT-INT ...)

; wrote /Users/linuxoid/Desktop/VUZICH/FP/lab2/./lab2_12.fasl
; compilation finished in 0:00:00.013
#P"/Users/linuxoid/Desktop/VUZICH/FP/lab2/lab2_12.fasl"
NIL
NIL
* (load "lab2_12.fasl")
T
* (count-int 2 '((1 2) 2 (4 2 3)))
3
* (count-int 2 '())
0
* (count-int -5 '((-5 4 (-5)) 2 (-5 (-1 (-5)) 3)))
4
* (count-int 0 '((5 0 (0 0 0) (-7 8 (9))) 23 (-2 (0 (-5 (434 (2
   0)))) 3)))
6
* (count-int 7 '((4 (5 300 134) 244 (1010) ) 2 (442 2 (43 (2 1 1
   1)) 3)))
0
* (count-int 7 '(7 2 2 (4 2 3)))
1
* (count-int -79 '(5 41 434 ( (6 -79) (4 2 -79 3) )))
2
* ^Z
[5]+  Stopped                 sbcl
(base) MacBook-Pro-Lina:lab2 linuxoid$
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Было интересно поработать с рекурсией в `Common lisp`.

11. Выводы

Во время выполнения данной лабораторной я познакомилась с функциями обработки списков: `car`, `cdr`. Реализовала древовидную рекурсию для обхода. Применила `cond`-выражения в своем алгоритме.