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

Студент группы М8О-307-19 МАИ *Инютин Максим Андреевич*, №10 по списку
Контакты: mainyutin@gmail.com
Работа выполнена: 24.03.2022

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

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

3. Задание (вариант № 2.35)

Запрограммируйте рекурсивно на языке Common Lisp функцию, вычисляющую множество всех подмножеств своего аргумента.

Исходное множество представляется списком его элементов без повторений, а множество подмножеств — списком списков.

4. Оборудование студента

Процессор Intel Core i7-9750H (12) @ 4.5GHz, память: 32 Gb, разрядность системы: 64.

5. Программное обеспечение

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

6. Идея, метод, алгоритм

Буду хранить текущее подмножество в переменной tmp, а множество всех подмножеств в res. На текущем шаге рекурсии проверяю, пустой ли список lst. Если так, то добавляю в res текущее подмножество tmp, иначе вызову эту же функцию от хвоста lst, добавляя и не добавляя голову в конец tmp.

Алгоритм использует древовидную рекурсию. На каждом шаге делается ещё два вызова функции, а $append$ двух списков имеет линейную сложность, поэтому асимптотика $O(2^n \cdot n)$, где n — длинна исходного списка.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
; 2.35

(defun subsets (lst)
        (subsets_recursive lst '() '()))

(defun subsets_recursive (lst res tmp)
        (if (null lst)
                (append res (list tmp))
                (append
                (subsets_recursive (rest lst) res tmp)
                (subsets_recursive (rest lst) res (append tmp
   (list (car lst)))))))
```

8.2. Результаты работы

```
[1]> (load "lab2.lisp")
;; Loading file lab2.lisp ...
;; Loaded file lab2.lisp
#P"/home/engineerxl/Study/6
   term/functional-programming/lab2/lab2.lisp"
[2]> (subsets '())
(NIL)
[3]> (subsets '(1))
(NIL (1))
[4]> (subsets '(1 2))
(NIL (2) (1) (1 2))
[5]> (subsets '(2 1))
(NIL (1) (2) (2 1))
[6]> (subsets '(1 2 3))
(NIL (3) (2) (2 3) (1) (1 3) (1 2) (1 2 3))
[7]> (subsets '(3 2 1))
(NIL (1) (2) (2 1) (3) (3 1) (3 2) (3 2 1))
[8]> (subsets '(4 2 3 1))
(NIL (1) (3) (3 1) (2) (2 1) (2 3) (2 3 1) (4) (4 1) (4 3) (4 3
   1) (4 2) (4 2 1) (4 2 3) (4 2 3 1))
[9]> (subsets (list 1 (list 2 "a" 3) "b"))
(NIL ("b") ((2 "a" 3)) ((2 "a" 3) "b") (1) (1 "b") (1 (2 "a" 3))
   (1 (2 "a" 3) "b"))
[10]> (subsets (list (list "another" "list") "element"))
(NIL ("element") (("another" "list")) (("another" "list")
   "element"))
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Можно уменьшить сложность алгоритма до $O(2^n)$, если не хранить множество всех множеств, а сразу печатать подмножества, но в лабораторной работе требуется вернуть список списков.

11. Выводы

Я познакомился со списками в Common Lisp. Работа с ними очень похожа на работу со списками в логическом языке, поэтому было несложно выполнить работу.