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

Студент группы 8О-307 Хренов Геннадий, № по списку 23.

Контакты: khrenov.gena@yandex.ru
Работа выполнена: 15.03.2021
Преподаватель: Дмитрий Анатольевич Иванов, 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 № 2.35)

Вариант 2.35 (сложность 3)

Запрограммируйте рекурсивно на языке Коммон Лисп функцию, вычисляющую множество всех подмножеств своего аргумента.

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

Примеры

```
(subsets '(1 2 3))
=> (() (1) (2) (3) (1 2) (1 3) (2 3) (1 2 3))
```

4. Оборудование студента

Ноутбук ASUS TUF GAMING, процессор AMD Ryzen 7 3750H 2.30GHz, память 8ГБ, 64-разрядная система.

5. Программное обеспечение

ОС Windows 10, программа LispWorks Personal Edition 6.1.1

6. Идея, метод, алгоритм

Алгоритм заключается в следующем: запускаем рекурсию, которая каждый вызов будет отделять от списка голову и продолжать выполнение с концом списка. Останавливаемся, когда разберем весь список на элементы, далее начинается восходящий процесс, в котором и лежит основная идея. Каждый вызов рекурсии берет список и к каждому элементу присоединяет голову своего вызова. Полученный новый список объединяется с входным и

передается дальше. Таким образом, мы перебираем всевозможные варианты подмножеств данного множества.

7. Сценарий выполнения работы

Для начала необходимо изучить правила работы со списками в lisp. Обратиться к голове и хвосту списка можно с помощью селекторов first и rest. Соединить 2 или более списков можно с помощью функции append. Функционал mapcar позволяет накапливать и возвращать список результатов, полученных применением некоторой функции *f* к первым, вторым и т.д. элементам всех списков l_1, ..., l_n, полученных в качестве аргумента. Далее реализуем алгоритм с помощью рекурсии, позволяющий получить все подмножества множества.

8. Распечатка программы и её результаты

Программа

```
;;генерация всех подмножеств данного множества

(defun subsets (lst)
  (if lst
      (let ((head (first lst))
            (z (subsets (rest lst))))
       (append (mapcar (lambda (y) (cons head y)) z) z))

      '(NIL)))
```

Результаты

```
CL-USER 1 > (defun subsets (lst)
  (if lst
      (let ((head (first lst))                          ;отделяем голову
            (z (subsets (rest lst))))                   ;запуск рекурсии для хвоста
       (append (mapcar (lambda (y) (cons head y)) z) z))  ;к каждому элементу либо прибавляем  голову
                                                          ;либо нет, соединяем все в один список
      '(NIL)))
SUBSETS

CL-USER 2 > (subsets '())
(NIL)

CL-USER 3 > (subsets '(4))
((4) NIL)

CL-USER 4 > (subsets '(2/3 3 9))
((2/3 3 9) (2/3 3) (2/3 9) (2/3) (3 9) (3) (9) NIL)

CL-USER 5 > (subsets '(a b c d))
((A B C D) (A B C) (A B D) (A B) (A C D) (A C) (A D) (A) (B C D) (B C) (B D) (B) (C D) (C) (D) NIL)

CL-USER 6 >
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание
1				

10. Замечания автора по существу работы

Обычно задача по поиску всех подмножеств заключается в генерации всех возможных сочетаний элементов данного множества. Однако благодаря функционалу Коммон Лисп

задачу можно решать исходя из логических соображений, которые более интуитивно понятны для человека.

11. Выводы

В данной лабораторной работе я научился работать со списками в Коммон Лисп. Этот тип данных имеет здесь довольно мощный функционал, и работать со списками можно просто и без проблем. Знания таких функционалов позволяет значительно уменьшить длину кода.