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

Студент группы 8О-308 Коростелев Дмитрий, № по списку 11.

Контакты: dmitry.k48@yandex.ru
Работа выполнена: 12.03.2021
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

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

3. Задание (вариант № 2.35/3)

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.

4. Оборудование студента

Ноутбук Asus VivoBook Pro 15, процессор Intel® Core™ i7-7700HQ CPU 2.80GHz 2.81GHz, память 8ГБ, 64-разрядная система.

5. Программное обеспечение

ОС Windows 10, программа LispWorks Personal Edition 7.1.2

6. Идея, метод, алгоритм

Для того, чтобы получить все подмножества заданного множества, нужно выбрав элемент списка, создать новый список с этим элементом и без него и запустить рекурсивно такой же поиск от двух новых списков. Удобно будет разделить заданный на список на два подсписка – левая и правая части и работать с первым элементом правой части. Таким образом рекурсивный алгоритм сводится к следующему – так как в подмножество заданного множества входит и сам список мы делаем конкатенацию левой и правой части и добавляем ее к ответу, если ранее его не было, далее добавляем к ответу все подмножества текущего списка, передвинув первый элемент правой части в конец левой ([левая часть + первый

элемент правой части] + [хвост правой части]) и все подмножества списка без первого элемента правой части ([левая часть] + [хвост правой части]).

Важное замечание: чтобы избавится от повторений, следует помнить, что если мы не убираем никакой элемент из списка, а простой делаем его сдвиг из одной части в другую, то и на следующем шаге нам не нужно выводить итоговый список, так как он был получен на предыдущим именно для этого и была введена третья переменная p в функции ss-ht, которая и выполняет данную роль.

7. Сценарий выполнения работы

Написать функцию ss-ht и убедится в ее корректности

Написать функцию subsets из которой будет запускаться ss-ht от заданного списка

8. Распечатка программы и её результаты

Программа

```
(defun subsets (lst)
     (ss-ht (list) lst T)     ;lst -> nil | lst
)
(defun ss-ht(head tail p)
     (cond
          ;условие окончания рекурсии
          ((null tail)
               (if p (list head))
          )
          (t (append
          ;head + first tail + rest tail = head + tail
          ;передаем nil в третий аргумент, чтобы избежать повтора
          (ss-ht (append head (list (first tail))) (rest tail) nil)
          ;head + rest tail != head + tail
          ;новый список, выводим его, поэтому T
          (ss-ht head (rest tail) T)
          ;секция с выводом списка
          (if p (list (append head tail)))
          )
          )
     )
)
```

Результаты

```
CL-USER 1 > (subsets '())
(NIL)


CL-USER 2 > (subsets '(1))
(NIL (1))


CL-USER 3 > (subsets '(1 2))
((1) NIL (2) (1 2))

CL-USER 4 > (subsets '(1 2 3))

((1 2) (1) (1 3) (2) NIL (3) (2 3) (1 2 3))

CL-USER 5 > (subsets '(1 2 3 4))

((1 2 3) (1 2) (1 2 4) (1 3) (1) (1 4) (1 3 4) (2 3) (2) (2 4) (3)
NIL (4) (3 4) (2 3 4) (1 2 3 4))
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Много времени ушло на то, чтобы понять, как вывести в консоль пустой список в виде двух скобок, а не NIL. Как оказалось NIL и пустой список – эквивалентные понятия и работа по итогу выполнена правильно.

11. Выводы

Написал рекурсивную функцию по поиску всех подмножеств заданного множества, протестировал ее и убедился в корректности ее работы, научился создавать списки, изучил базовые функции работы со списками: получение элемента в списке, конкатенация двух списков, создание списка из элементов и т.д.