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

Студент группы 8О-307 МАИ *Бронников Максим*, №4 по списку
Контакты: max120199@gmail.com
Работа выполнена: 01.04.2020

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 №35)

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 ПЭВМ студента

Процессор Intel© Celeron© CPU @ 2.16GHz x 2, память: 4096Mb, разрядность системы: 64.

5. Программное обеспечение ЭВМ студента

OS Linux Mint 19.3 Cinnamon, среда SLIME 2.24 с реализацией языка SBCL 1.4.5.debian

6. Идея, метод, алгоритм

Функция `subsets` принимает в качестве аргумента список и работает следующим образом:

1. Проверяет является ли пустым переданный список, если да, то возвращает список с пустым списком внутри.
2. С помощью вспомогательной функции `concat` возвращает объединение содержимого 2-ух списков которые получаются как:
   - Результату рекурсивного приминения функции `subsets` к хвосту списка - аргумента.
   - Применения вспомогательной функции `attach` к голове списка - аргумента и результату приминения функции `subsets` к хвосту списка. Вспомогательная функция `attach` добавляет элемент к каждому списку, содержащемуся в списке, который передается ей в качестве аргумента.

Функция `concat` рекурсивно спускается, пока первый переданный аргумент не *nil*, после чего начинает последовательно присоединять к результату элементы из первого аргумента от последнего к первому, постепенно выходя из рекурсии.

Функция `attach` рекурсивно присоедениняет результат слияния переданного элемента с головой переданного списка к результату применения `attach` к хвосту переданного списка.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
;;; define function of concatination 2 list in 1
(defun concat (set1 set2)
    (if (null set1) set2 (cons (car set1) (concat (cdr set1)
   set2))))

;;; function append element in head of all list in list
(defun attach (elem set)
    (if (null set) nil
    (cons (cons elem (car set)) (attach elem (cdr set)))))

;;; main function
(defun subsets (set)
```

```
    ;; if set is empty => return (())
    (if (null set) '(nil)
    ;; else concat
    (concat
    ;; and subset of tail
    (subsets (cdr set))
    ;; subset of tail with head
    (attach (car set) (subsets (cdr set))))))
```

8.2. Результаты работы

```
(base) max@max-Lenovo-B50-30:~/FuncProg/lab2$ ls
main.lisp
(base) max@max-Lenovo-B50-30:~/FuncProg/lab2$ sbcl
This is SBCL 1.4.5.debian, 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 "main.lisp")

; compiling file "/home/max/FuncProg/lab2/main.lisp" (written 01
   APR 2020 01:49:46 PM):
; compiling (DEFUN CONCAT ...)
; compiling (DEFUN ATTACH ...)
; compiling (DEFUN SUBSETS ...)

; /home/max/FuncProg/lab2/main.fasl written
; compilation finished in 0:00:00.020
#P"/home/max/FuncProg/lab2/main.fasl"
NIL
NIL
* (load "main.fasl")

T
* (subsets '(1 2 3))

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

```
* (subsets '())

(NIL)
* (subsets '(a))

(NIL (A))
* (subsets '(a b))

(NIL (B) (A) (A B))
*
(base) max@max-Lenovo-B50-30:~/FuncProg/lab2$ ls
main.fasl   main.lisp
(base) max@max-Lenovo-B50-30:~/FuncProg/lab2$ exit
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

В демонстрации работы я не включил тесты с большей длинной списка, поскольку размер результирующего списка будет 2^n, где n- размер входного списка, что порождает громозкий ответ в листинге.

11. Выводы

В процессе выполнения работы я познакомился с основным типом данных в *Common Lisp* - списком, и даже написал простейшую программу с использованием стандартных функций работы со списками.

Все вспомогательные функции работают за линейное время, поскольку делает пропорциональное размеру аргументов количество рекурсивных вызовов, на каждом из которых делает действия с константным временем исполнения.
Основная же функция работает за время $O(n2^n)$, поскольку на каждом этапе делает добавление и склейку линейной сложности(см. выше) с 2^n элементов, а также вызывает в себе 2 раза рекурсивно себя же, но со списком длины $(n-1)$.

Эта задача заставила меня поломать голову, поскольку мне было непривычно перестроится на ее выполнение после выполнения лабораторой работы по другому предмету на языке *Python*, однако я считаю полезной такую разминку мозга и надеюсь на не менее трудные задания в дальнейшем.