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

Студент группы 8О-307 МАИ *Ефимов Александр*, №7 по списку
Контакты: aleks.efimov2011@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

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

3. Задание

Вариант: №2.36
Дан список действительных чисел $(X_1...X_n)$. Запрограммируйте рекурсивно на языке Common Lisp функцию, которая возвращает:

- сам список, если последовательность $X_1, ..., X_n$ упорядочена по убыванию, т.е. $X_1 > X_2 > ... > X_n$;
- список $(X_n...X_1)$ в противном случае.

4. Оборудование студента

Процессор Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, память: 7.6Gi, разрядность системы: 64.

5. Программное обеспечение

ОС Arch Linux, система CLisp.

6. Идея, метод, алгоритм

Проверить, если значения в списке убывают. Причем, если размер списка меньше двух, считать, что он убывающий.

Вернуть его же, если он убывающий, или обратить иначе.

7. Распечатка программы и её результаты

7.1. Исходный код

```
;;:: Checks if given list is descending and reverses it
;;;; if it is not

(defun is-descending (l)
  "Checks if given list is in descending order"
  (cond ((< (length l) 2) T)
        ((<= (first l) (second l)) NIL)
        (T (is-descending (rest l)))))

(defun decreasing (l)
  "Reverses list if it is not in non-ascending order"
  (if (is-descending l)
      l
      (reverse l)))
```

7.2. Результаты работы

```
[1]> (load "descend-reverse.lisp")
;; Loading file descend-reverse.lisp ...
;; Loaded file descend-reverse.lisp
[2]> (decreasing (list 1))
(1)
[3]> (decreasing (list 1 2))
(2 1)
[4]> (decreasing (list 3 2 1))
(3 2 1)
[5]> (decreasing (list 3 3 2))
(2 3 3)
[6]> (decreasing (list 1 2 4 3))
(3 4 2 1)
[7]> (decreasing (list 9 8 7 6 5 1))
(9 8 7 6 5 1)
```

```
[8]> (decreasing (list 9 7 1 5))
(5 1 7 9)
```

---

8. Выводы

По спискам можно итерировать с помощью функций *first* и *rest*.