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

Студент группы М8О-307 МАИ *Днепров Иван*, №10 по списку
Контакты: vanya.dneprov@gmail.com
Работа выполнена: 17.03.2020

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 №2.36)

Дан список действительных чисел (X1 ... Xn).

Запрограммируйте рекурсивно на языке Коммон Лисп функцию, которая возвращает

сам список, если последовательность X1, ..., Xn упорядочена по убыванию, т.е. X1 > X2 > ... > Xn; список (Xn ... X1) в противном случае.

4. Оборудование студента

MacBook (13-inch, Mid 2010), процессор 2,4 GHz Intel Core 2 Duo, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

Mac OS 10.13.6, компилятор clisp, текстовый редактор Sublime Text 3.

6. Идея, метод, алгоритм

Для простоты я решил разбить задачу на две подзадачи. Первая задача – реверс списка (функция reverse-list). Вторая – проверка списка на убывание (функция is-decreasing).

Принцип работы reverse-list заключается в том, что на каждой итерации эта функция выдергивает первый элемент из списка в первом параметре функции и вставляет его в конец списка во втором необязательном параметре. Функция прекращает работу когда первый параметр становится пустым списком.

Принцип работы is-decreasing заключается в том, что эта функция сравнивает первый и второй элементы списка, отсекая от списка по элементу за итерацию. Если попадается элемент <= следующему, функция возвращает nil, а если список стал пустым (прошли все элементы) – T.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun reverse-list (w &optional acc)
  (if w (reverse-list (cdr w) (cons (car w) acc)) acc))

(defun is-decreasing (w)
  (cond ((null (cdr w)) t)
    ((<= (car w) (cadr w)) nil)
    ((is-decreasing (cdr w)))))

(defun decreasing (l)
  (if (is-decreasing l)
    l
    (reverse-list l)))
```

8.2. Результаты работы

```
[4]> (decreasing '(9 8 7 6 5 1))
(9 8 7 6 5 1)
[5]> (decreasing '(9 7 1 5))
(5 1 7 9)
[6]> (decreasing '(9 8 7 6 5 1 1))
(1 1 5 6 7 8 9)
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Я и в прошлой лабораторной работе использовал списки, по этому вторая это задание не показалось мне сложным.

11. Выводы

Вообще, списки – мощьнейший инструмент в функциональном программировании, а учитывая, сколько в Lisp есть интересных функций по работе со списками, думаю, что большинство функций, написанных на Lisp так или иначе связанны с этой структурой данных.