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

Студент группы 8О-307 МАИ *Спиридонов Кирилл*, №18 по списку
Контакты: vo-ro@list.ru
Работа выполнена: 02.04.22

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

Освоить списки в Коммон Лисп. Операции над ними. Применить селекторы. Научиться пользоваться map-функционалом. Воспользоваться лямбда-выражениями.

3. Задание (вариант №2.9)

Дан список действительных чисел $(x_1, ..., x_n), n \geq 2$.
Запрограммируйте рекурсивно на языке Коммон Лисп функцию, вычисляющую выражение вида:
$(x_1 * x_n) + (x_2 * x_{n-1}) + ... + (x_n * x_1)$.

Примеры
(sum-product2 '(1 2 3 4 5)) =>
(1*5) + (2*4) + (3*3) + (4*2) + (5*1) =>
5 + 8 + 9 + 8 + 5 => 35

4. Оборудование студента

Процессор Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, память: 8192Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 20.04 LTS, среда LispWorks Personal Edition 7.1.2

6. Идея, метод, алгоритм

Идея алгоритма простая. Имеется исходная функция `sum-product2 (l)`, которая принимает список и возращает сумму элементов, попарно перемноженных соответствующих элементов исходного списка с перевернутым этим же списком. Переворачивание списка осуществляется при помощи встроенной функции `reverse (l)`. Для поэлементного умножения воспользуемся лямбда-выражением `(lambda (x1 x2) (* x1 x2))`. В итоге получим список, который состоит из перемноженных соответствующих элементов исходного списка с перевернутым этим же списком. Для него вызываем функцию `sum-of-list (l)`, которая возращает сумму всех элементов списка. Полученная сумма и будет ответом на задачу.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun extra-fun (res l)
  (if (rest l)
      (extra-fun (+ res (first l)) (rest l))
    (+ res (first l))))

(defun sum-of-list (l)
  (extra-fun 0 l))

(defun sum-product2 (l) (sum-of-list (mapcar (lambda (x1 x2) (*
   x1 x2)) l (reverse l))))
```

8.2. Результаты работы

CL-USER 4 > (sum-product2 (list 1 2 3 4 5))
35
CL-USER 5 > (sum-product2 (list 3 5 8 9 1))
160

CL-USER 6 > (sum-product2 (list 3 4 4 3))
50
CL-USER 7 > (sum-product2 (list 5 0 1 -2 3))
31

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Программа работает за O(n). Можно было бы перемножать попарно не все элементы, а только половину и затем результат умножать на 2. Но на асимптотику это бы не повлияло.

11. Выводы

В ходе выполнения лабораторной работы я познакомился с типом данных лист. Узнал как он устроен (голова + хвост). Применил лямбда-выражение. Понял для чего оно нужно, а именно, для сокращения кода и лучшей читабельности программы. Приобрёл навыки обрабатывания списков.