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

Студент группы 8О-307 МАИ *Бирюков Виктор*, №2 по списку
Контакты: vikvladbir@mail.ru
Работа выполнена: 26.03.2022

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 $(X_1 \dots X_n)$. Запрограммируйте рекурсивно на языке Коммон Лисп функцию, которая возвращает

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

4. Оборудование студента

Процессор AMD Ryzen 7 3700U @ 2.3GHz, память: 20Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Windows 10, компилятор SBCL 2.2.2, текстовый редактор Sublime Text 4.

6. Идея, метод, алгоритм

Функция `myreverse` осуществляет обращение списка, аналогично встроенной функции `reverse`. Вспомогательная функция `reverse-iter` описывает линейно-итеративный процесс: пока первый аргумент не пуст, его первый элемент ставится в начало второго аргумента, затем функция вызывается рекурсивно от хвоста первого аргумента и увеличенного второго. Когда первый список опустеет, второй будет содержать его в перевернутом виде.

Функция `decrease-p` проверяет, что элементы списка упорядочены по убыванию. Функция описывает линейно-итеративный процесс: список из менее двух элементов считается упорядоченным, иначе проверяется упорядоченность первых двух элементов, затем рекурсивно — упорядоченность списка без первого элемента.

Функция `decreasing` проверяет упорядоченность списка функцией `decrease-p` и, если его элементы не убывают, переворачивает список функцией `myreverse`, иначе возвращает исходный.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
; 2.36

(defun reverse-iter (a b)
        (if (null a)
                b
                (reverse-iter (rest a) (cons (first a) b))))

(defun myreverse (a)
        (reverse-iter a NIL))

(defun decrease-p (l)
        (if (null (rest l))
                T
                (if (> (first l) (first (rest l)))
                        (decrease-p (rest l))
                        NIL)))

(defun decreasing (l)
        (if (decrease-p l)
                l
```

```
                (myreverse l)))
```

8.2. Результаты работы

```
* (decreasing (list 9 8 7 6 5 1))
(9 8 7 6 5 1)

* (decreasing (list 9 7 1 5))
(5 1 7 9)

* (decreasing (list 3 3 3 2 1))
(1 2 3 3 3)

* (decreasing (list 1))
(1)

* (decreasing ())
NIL
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Временная сложность функции — $O(n)$, в худшем случае — $O(2n)$, где n — длина списка. Если объединить первую и вторую функции, можно уменьшить коэффициент и получить сложность $O(n)$ во всех случаях за счет увеличения пространственной сложности до $O(3n)$, так как придется дополнительно хранить исходный список.

11. Выводы

В ходе выполнения лабораторной работы я познакомился со встроенной структурой данных список. Как и во многих других неимперативных языках, в Коммон Лисп списки имеют структуру, схожую со стековой, что упрощает сложность некоторых операций до константной, сложность же других операций становится линейной.