**Министерство науки и высшего образования РФ**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский авиационный институт»**

**Национальный исследовательский университет**

**Институт** №8: «Информационные технологии и прикладная математика»

**Кафедра** 806: «Вычислительная математика и программирование»

**Лабораторная работа №2**

**по курсу «Функциональное программирование»**

Студент: Назарова А.И.

Преподаватель: Иванов Д.А.

Группа: М80-308Б-19

Дата:

Оценка:

Подпись:

Москва, 2022

# Тема работы

Простейшие функции работы со списками Коммон Лисп.

# Цель работы

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

# Задание (вариант 2.41)

Запрограммируйте рекурсивно на языке Коммон Лисп функцию-предикат tree-similar-p (x y), которая принимает два аргумента - дерева, представленных в виде списков атомов. Предикат должен вернуть истину, если одинаковые атомы расположены в списках х и у в одном и том же порядке при обходе дерева слева направо, т.е. независимо от внутренней структуры х и у.

# Оборудование студента

Ноутбук Xiaomi, процессор: Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 2001 МГц, ядер: 4, логических процессоров: 8

# Программное обеспечение

Майкрософт Windows 10 Домашняя

# Идея, метод, алгоритм

Так как в задании не нужно учитывать структуру дерева, то можно заметить, что достаточно проверить на эквивалентность списки, представляющие деревья. Но так как списки могут быть вложенными, сначала необходимо избавиться от вложенности. Функция to-list раскрывает все вложенные списки, а функция equal проверяет эквивалентность двух списков.

# Сценарий выполнения работы

# Распечатка программы и ее результат

Программа

(defun to-list (tree)  
 (if (null tree)  
 nil  
 (if (atom (first tree))  
 (cons (first tree) (to-list (rest tree)))  
 (append (to-list (first tree)) (to-list (rest tree)))  
 )  
 )  
)

(defun tree-similar-p (tree1 tree2)  
 (equal (to-list tree1) (to-list tree2))  
)

Результат

> (tree-similar-p '(1 (2 (3 4)) 5) '((1 2) 3 (4 5)))  
T  
> (tree-similar-p '(1 (2 (3 4)) 5) '((1 2) 3 (5 4)))  
NIL  
> (tree-similar-p '((1 2)) '(1 2))  
T  
> (tree-similar-p '((1) (2)) '((1) 2))  
T  
> (tree-similar-p '(1 (2 (3))) '(((3) 2 ) 1))  
NIL

# Дневник отладки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Дата, время | Событие | Действие по исправлению | Примечание |
|  |  |  |  |  |

# Замечания автора по существу работы

Работа показалось слишком простой с точки зрения логики.

# Выводы

Выполняя эту лабораторную работу, я познакомилась с новыми для меня возможностями языка Common Lisp, получила навыки работы со списками. Списки в Lisp'е довольно мощное средство, и в то же время просты в изучении. С их помощью можно удобно представлять деревья, очень важную структуру данных.