**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 №1**

**по курсу «Функциональное программирование»**

Студент: Назарова А.И.

Преподаватель: Иванов Д.А.

Группа: М80-308Б-19

Дата:

Оценка:

Подпись:

Москва, 2022

# Тема

Примитивные функции и особые операторы Коммон Лисп.

# Цель работы

Научиться вводить S-выражения в Лисп-систему, определять переменные и функции, работать с условными операторами, работать с числами, используя схему линейной и древовидной рекурсии.

# Задание (вариант 1.33)

Поле шахматной доски определяется парой натуральных чисел, каждое из которых не превосходит восьми:

* первое число - номер вертикали (при счете слева направо).
* второе - номер горизонтали (при счете снизу вверх),

Определите на языке Коммон Лисп функцию c четырьмя параметрами - натуральными числам k, l, m, n, каждое из которых не превосходит восьми.

k, l - Задают поле, на котором расположена фигура - слон.

m, n - Задают поле, куда он должен попасть.

Функция должна возвращать:

- T, если слон (k,l) может попасть на поле (m,n) за один ход;

- i, j - два значения с помощью values, если слон (k,l) может попасть на поле (m,n) за два хода через поле (i,j);

- NIL, если слон не может перейти на указанное поле (поля разного цвета).

# Оборудование студента

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

# Идея, метод, алгоритм

Слон может ходить только по диагонали, поэтому за один шаг он может попасть только в клетку того же цвета, которая расположена на любой диагонали, пересекающейся с текущей клеткой(позицией) слона. Затем, чтобы вычислить клетку, через которую слон может попасть в итоговую, занимаемся составлением уравнений. Путем долгих вычислений, мы их находим. Затем делаем проверку, чтобы координаты этих точек не были отрицательными (т.е. лежали внутри поля)

# Исходный код

(defun one-step(k l m n)  
 (cond ((or (= (- k m) (- l n)))t)  
 ((or (= (- k m) (- n l)))t)  
 ))

(defun two-steps (k l m n)  
 (let\* (  
 (i1 (/ (+ (+ k m) (- n l))2) )  
 (j1 (/ (+ (- m k) (+ n l))2) )  
 (i2 (/ (+ (+ k m) (- l n))2) )  
 (j2 (/ (+ (- k m) (+ n l))2) )  
 )

(if (and (> i1 0 ) (> j1 0 )) (list i1 j1) (list i2 j2))  
 )  
)

(defun bishop-moves(k l m n)  
 (cond  
 ((one-step k l m n)t)  
 ((/= ( rem (+ k l) 2) (rem (+ m n) 2)) nil)  
 (T (values-list (two-steps k l m n)))  
))

# Результат работы программы

(bishop-moves 4 4 7 7)

T

(bishop-moves 1 1 2 6)

4  
4

(bishop-moves 1 1 1 2)

NIL

(bishop-moves 1 1 2 7)

NIL

# Выводы

Данная лабораторная работа сначала показалось слишком простой, но в итоге пришлось повозиться с выводом клетки(необходимо было составить несколько уравнений, выразить координаты точек и т.д.). Также при выполнении данной лабораторной работы я вспомнила синтаксис языка.