Отчет по лабораторной работе № 1 по курсу «Функциональное программирование»

Студент группы 8О-307 МАИ *Спиридонов Кирилл*, №18 по списку
Контакты: `vo-ro@list.ru`
Работа выполнена: 21.03.22

Преподаватель: Иванов Дмитрий Анатольевич, доц. каф. 806
Отчет сдан:
Итоговая оценка:
Подпись преподавателя:

1. Тема работы

Примитивные функции и особые операторы Коммон Лисп

2. Цель работы

Научиться пользоваться базовыми операторами такими как: if, cond. Строить логические выражения с помощью булевых операций AND, OR, NOT. Определять функции и вызывать их в Лисп-системе.

3. Задание (вариант №1.32)

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 с четырьмя параметрами — натуральными числам `k, l, m, n`, каждое из которых не превосходит восьми.

`k, l`

Задают поле, на котором расположена фигура - конь.

`m, n`

Задают поле, куда он должен попасть.

Функция должна возвращать:

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

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

`NIL`
если конь не может перейти на указанное поле ни за одни, ни за два хода.

Примеры
(knight-moves 1 1 3 2) => T
(knight-moves 1 1 5 3) => 3, 2
(knight-moves 1 1 8 8) => NIL

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 вспомногательная функция `check-next (k l m n)`, которая возращает `T`, если из текущей позиции `(k, l)` можно достич позиции `(m, n)`. Основная функция `knight-moves (k l m n)` принимает начальные данные и проверяет при помощи `check-next` можно ли за один ход достич `(m, n)`, если да, то возращается `T`. Иначе проходимся по остальным позициям так, будто бы уже был сделан шаг. И если функция `check-next` вернёт `T`, то возращаем пару `i, j` откуда был сделан шаг. Если не найдётся таких клеток, вернётся `NIL`.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun check-next (k l m n)
  (cond ((and (= (+ 2 k) m) (= (1+ l) n)) T)
        ((and (= (+ 2 k) m) (= (- l 1) n)) T)
        ((and (= (- k 2) m) (= (- l 1) n)) T)
        ((and (= (- k 2) m) (= (- l 1) n)) T)
        ((and (= (1+ k) m) (= (+ 2 l) n)) T)
        ((and (= (1+ k) m) (= (- l 2) n)) T)
        ((and (= (- k 1) m) (= (+ 2 l) n)) T)
        ((and (= (- k 1) m) (= (- l 2) n)) T)))
 (defun knight-moves (k l m n)
  (cond ((check-next k l m n) T)
        ((AND (<= (+ 2 k) 8) (<= (1+ l) 8) (check-next (+ 2 k)
   (1+ l) m n)) (values (+ 2 k) (1+ l)))
        ((AND (<= (+ 2 k) 8) (>= (- l 1) 1) (check-next (+ 2 k)
   (- l 1) m n)) (values (+ 2 k) (- l 1)))
        ((AND (>= (- k 2) 1) (>= (- l 1) 1) (check-next (- k 2)
   (- l 1) m n)) (values (- k 2) (1- l)))
        ((AND (>= (- k 2) 1) (<= (1+ l) 8) (check-next (- k 2)
   (1+ l) m n)) (values (- k 2) (1+ l)))
        ((AND (<= (1+ k) 8) (<= (+ 2 l) 8) (check-next (1+ k) (+
   2 l) m n)) (values (1+ k) (+ 2 l)))
        ((AND (<= (1+ k) 8) (>= (- l 2) 1) (check-next (1+ k) (-
   l 2) m n)) (values (1+ k) (- l 2)))
        ((AND (>= (- k 1) 1) (<= (+ 2 l) 8) (check-next (- k 1)
   (+ 2 l) m n)) (values (- k 1) (+ 2 l)))
        ((AND (>= (- k 1) 1) (>= (- l 2) 1) (check-next (- k 1)
   (- l 2) m n)) (values (- k 1) (- l 2)))))
```

8.2. Результаты работы

```
   CL-USER 2 > (knight-moves 1 1 2 3)
T
CL-USER 3 > (knight-moves 2 2 3 3)
4
1
CL-USER 4 > (knight-moves 1 1 8 8)
NIL
```

```
CL-USER 5 > (knight-moves 3 3 3 3)
5
4
CL-USER 6 > (knight-moves 6 6 4 3)
NIL
CL-USER 7 > (knight-moves 7 7 6 5)
T
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Изначально хотел реализовать рекурсивно, использова доболнительную функцию, которая считает глубину. Но были проблемы с выводом результата. Провозился часов 6 и ни к чему не пришёл. Рекурсия здесь выглядила бы лучше. Но т.к. считается всего два шага, то можно можно как сделал я. Наметки для рекурсии:

```
(defun knight-moves (k l m n)
  (extra-fun k l m n 0))
;d - depth
(defun extra-fun (k l m n d)
  (cond ((OR (= d 3) (> k 8) (> l 8) (< k 1) (< l 1))
  NIL)
        ((AND (= k m) (= l n) (/= d 0)) T)
        ((cond ((extra-fun (+ 2 k) (+ 1 l) m n (+ 1 d))
  T))
    .....
```

Программа работает за константное время O(1). Данная программа работает только для текущей задачи, т.е. просмотр на 1 или 2 шага вперёд.

11. Выводы

В ходе выполнения лабораторной работы я познакомился с основами языка CommonLisp: создавать и использовать функции, пользоваться условными операторами.