Отчет по лабораторной работе № 3
по курсу «Функциональное программирование»

Студент группы М8О-307-19 МАИ *Ежов Никита Павлович*, №9 по списку
Контакты: nikita.ejov2012@yandex.ru
Работа выполнена: 08.05.2022

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

1. Тема работы

Последовательности, массивы и управляющие конструкции Common Lisp.

2. Цель работы

Научиться создавать векторы и массивы для представления матриц, освоить общие функции работы с последовательностями, инструкции цикла и нелокального выхода.

3. Задание (вариант № 3.20)

Запрограммировать на языке Коммон Лисп функцию, принимающую в качестве аргумента двумерный массив, представляющий действительную матрицу произвольного размера.

Функция должна возвращать новую матрицу того же размера, получающуюся из данной перестановкой строк - первой с последней, второй с предпоследней и т.д.

Исходный массив должен оставаться неизменным.

4. Оборудование студента

Процессор Intel i7-4770 (8) @ 3.9GHz, память: 16 Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Kubuntu 20.04.4 LTS, комилятор GNU CLISP 2.49.92, текстовый редактор Visual Studio Code 1.67.1

6. Идея, метод, алгоритм

Скопирую матрицу, чтобы не менять исходную. Затем сохраню в отдельные переменные кол-во столбцов и строки скопированной матрицы.

Обход выполняется следующим образом: мы проходим по всем строчкам до $n1//2$ (целочисленное деление) и меняем элементы i-й строки с элементами $m1 - i$ строки. В программе обход реализован следующим образом: если $mod(m1, 2) == 1$, то мы отнимаем от $m1$ единицу, чтобы оно стало чётным числом. Потом запескаем цикл по строкам от нулевой и до $(m1/2) - 1$, меняя строки местами поэлементно, как это описано выше. Таким образом и реализована перестановка первой строки с последней, второй с предпоследней и т.д.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
; 3.20

; Returns matrix copy
(defun copy-matrix (a)
        (let* ((n (array-dimension a 0)) (m (array-dimension a
   1))
        (res (make-array (list n m) :initial-element 0.0)))
        (loop for i from 0 to (- n 1)
                do (loop for j from 0 to (- m 1)
                        do (setf (aref res i j) (aref a i j))))
        res))

(defun swap-matrix (matr)
        (let* (
        (m1 (array-dimension matr 0))
        (m2 (array-dimension matr 1))
        (a (copy-matrix matr)))
        (if (= (mod m1 2) 1)
                (- m1 1))
        (loop for i from 0 to (- (/ m1 2) 1)
                do(loop for j from 0 to (- m2 1)
                        do (rotatef (aref a i j) (aref a (- (-
   m1 1) i) j))))
       a))
```

8.2. Результаты работы

```
;; Загружается файл lab3.lisp ...
;; Загружен файл lab3.lisp
[1]> (setq b (make-array '(4 2) :initial-contents '((1 2) (2 -3)
   (3 3) (4 10))))
#2A((1 2) (2 -3) (3 3) (4 10))
[2]> (swap-matrix b)
#2A((4 10) (3 3) (2 -3) (1 2))
[3]> b
#2A((1 2) (2 -3) (3 3) (4 10))
[4]> (setq a (make-array '(3 2) :initial-contents '((1 2) (2 -3)
   (3 3))))
#2A((1 2) (2 -3) (3 3))
[5]> (swap-matrix a)
#2A((3 3) (2 -3) (1 2))
[6]> a
#2A((1 2) (2 -3) (3 3))
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Сложность полученного решения $O(n \cdot m)$. Пространственная сложность тоже $O(n \cdot m)$, поэтому добиться лучшей асимптотики нельзя.

11. Выводы

Я познакомился с векторами и массивами в Common Lisp. Язык поддерживает императивную парадигму в отличии от, например, Prolog, на котором работа с матрицами очень неприятная и сложная.