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

Студент группы 8О-307 Алексюнина Юлия, № по списку 1.

Контакты: juvyjuli@gmail.com
Работа выполнена: 07.05.2020
Преподаватель: Иванов Дмитрий Анатольевич, доц. каф. 806
Отчет сдан:
Итоговая оценка:
Подпись преподавателя:

1. Тема работы

Последовательности, массивы и управляющие конструкции Коммон Лисп

2. Цель работы

Цель работы: научиться создавать векторы и массивы для представления матриц, освоить общие функции работы с последовательностями, инструкции цикла и нелокального выхода

3. Задание (вариант № 48)

Запрограммировать на языке Коммон Лисп функцию, принимающую в качестве единственного аргумента двумерный массив - действительную матрицу. Функция должна возвращать новую матрицу (исходный массив должен оставаться неизменным), полученную путём допустимых преобразований из исходной матрицы, и такую, чтобы один из элементов матрицы, обладающий наименьшим значением, располагался в левом нижнему углу матрицы.

```
(defun swap-min-to-bottom-left (a)
  ...)
```

4. Оборудование студента

Ноутбук HP, процессор Intel® Core™ i3-6006U CPU 2.00GHz 1.99GHz, память 4ГБ, 64-разрядная система.

5. Программное обеспечение

ОС Windows 10 версия 1903, программа LispWorks Personal Edition 6.1.1

6. Идея, метод, алгоритм

Сначала мы копируем матрицу, чтобы исходная осталась нетронутой, и работаем уже с ее копией. Ищем, на какой позиции находится минимальный элемент матрицы, чтобы запомнить ее, а далее переставляем строки и столбцы так, чтобы элемент матрицы с минимальным значением оказался в левом нижнем углу.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

Программа

```
(defun print-matrix (matrix &optional (chars 3) stream)
    (let ((*print-right-margin* (+ 6 (* (1+ chars)
                                        (array-dimension matrix
1)))))
        (pprint matrix stream)
        (values)))

(defun swap-rows (mtrx r1 r2)
    (loop with n = (array-dimension mtrx 1)
        for i upfrom 0 below n do
            (rotatef (aref mtrx r1 i) (aref mtrx r2 i)))
    mtrx)

(defun swap-cols (mtrx c1 c2)
    (loop with n = (array-dimension mtrx 0)
        for i upfrom 0 below n do
            (rotatef (aref mtrx i c1) (aref mtrx i c2)))
    mtrx)

(defun pos-of-min (mtrx)
    (let ((min-row 0) (min-column 0) (min-elem (aref mtrx 0 0)))
        (loop with n = (array-dimension mtrx 0)
            for i upfrom 0 below n do
            (loop with m = (array-dimension mtrx 1)
            for j upfrom 0 below m do
                (cond ((< (aref mtrx i j) min-elem)
                    (setf min-elem (aref mtrx i j) min-row i min-
column j)))))
        (values min-row min-column)))

(defun copy-array (array)
    (adjust-array array (array-dimensions array)))
```

```
(defun swap-min-to-bottom-left (mtrx)
    (let ((new-mtrx (copy-array mtrx)))
        (multiple-value-bind (i j) (pos-of-min mtrx)
            (swap-rows new-mtrx (- (array-dimension mtrx 1) 1) i)
            (swap-cols new-mtrx 0 j))
        new-mtrx))


(defvar *m* (make-array '(4 4) :initial-contents '((1 2 3 4)
                                                   (5 -6 7 8)
                                                   (9 0 9 2)
                                                   (3 4 5 6))))
(defvar *n* NIL)

(defvar *x* (make-array '(5 5) :initial-contents '((5 7 -9 13 6)
                                                   (56 8 -1 2 4)
                                                   (8 1 6 77 9)
                                                   (1 4 -2 16 -
32)
                                                   (5 3 2 5 9))))
(defvar *y* NIL)
```

Результаты

```
CL-USER 2 : 1 > print-matrix *m*

#2A((1 2 3 4)
    (5 0 7 8)
    (9 6 9 2)
    (3 4 5 6))

CL-USER 3 : 1 > print-matrix (setf *n* (swap-min-to-bottom-left
*m*))
```

```
#2A((2 1 3 4)
    (4 3 5 6)
    (6 9 9 2)
    (0 5 7 8))

CL-USER 3 : 1 > print-matrix *m*

#2A((1 2 3 4)
    (5 0 7 8)
    (9 6 9 2)
    (3 4 5 6))

CL-USER 4 : 1 > print-matrix *x*

#2A((5 7 -9 13 6)
    (56 8 -1 2 4)
    (8 1 6 77 9)
    (1 4 -2 16 -32)
    (5 3 2 5 9))

CL-USER 5 : 1 > print-matrix (setf *y* (swap-min-to-bottom-left
*x*))

#2A((6 7 -9 13 5)
    (4 8 -1 2 56)
    (9 1 6 77 8)
    (9 3 2 5 5)
    (-32 4 -2 16 1))

CL-USER 6 : 1 > print-matrix *x*

#2A((5 7 -9 13 6)
    (56 8 -1 2 4)
    (8 1 6 77 9)
    (1 4 -2 16 -32)
```

```
(5 3 2 5 9))
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Задача не вызвала у меня затруднений. В ходе решения была реализована функция *print-matrix* печати матрицы для её удобной визуализации, а также в программе заданы два набора тестовых данных.

11. Выводы

В третьей лабораторной работе по курсу «Функциональное программирование» я познакомилась с массивами, последовательностями и матрицами. Я освоила базовые навыки обработки этих крайне полезных в программировании структур данных. Также я продолжила углубленное знакомство с синтаксисом языка Common Lisp.