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

Студент группы 8О-307 МАИ *Спиридонов Кирилл*, №18 по списку
Контакты: vo-ro@list.ru
Работа выполнена: 16.04.22

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

1. Тема работы

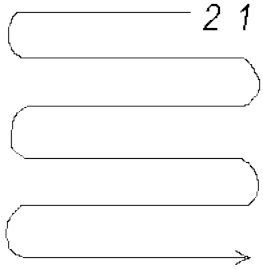
Последовательности, массивы и управляющие конструкции Коммон Лисп.

2. Цель работы

Освоить работу с массивами. Научиться пользоваться циклами.

3. Задание (вариант №3.43)

Запрограммировать на языке Коммон Лисп функцию, принимающую в качестве единственного аргумента целое число n - порядок матрицы. Функция должна создавать и возвращать двумерный массив, представляющий целочисленную квадратную матрицу порядка n, элементами которой являются числа $1, 2, ...n^2$, расположенные по схеме, показанной на рисунке.



Примеры
(matrix-tr-br 4) =>
#2A((4 3 2 1)
(5 6 7 8)
(12 11 10 9)

(13 14 15 16))

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. Функции `matrix-tr-br (n)`, на вход подаётся порядок матрицы. В этой функции создаём локальный двумерный массив. Затем циклом от 0 до n проходимся по всем строкам (на n+1-ой итерации возвращаем получившийся массив). Основная идея в том, что если мы находимся на строке с нечётным номером (строки номеруются с нуля), то будем заполнять строку слева направо в возрастающем порядке. Первый элемент в нечётной строке будет равен `n*i + 1` (`i -` номер строки), т.к. элемент над ним имеет значени `n*i`. В чётной строке мы также идём слева направо, но элементы уже убывают. Следуя такому алгоритму, получим массив необходимого вида.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun print-matrix (matrix &optional (chars 3) stream)
  ;; Предполагаем, что требуется
  ;;  3 знака по умолчанию на каждый элемент,
  ;;  6 знаков на #2A и скобки.
  (let ((*print-right-margin* (+ 6 (* (1+ chars) ; плюс пробел
                                      (array-dimension matrix
 1)))))
    (pprint matrix stream)
    (values)))
```

```
(defun matrix-tr-br (n)
  (let ((arr (make-array (list n n))))
    (loop for i from 0 to n do
        (when (= i n) (return arr))
        (if (oddp i)
            (loop for j from 1 to n do
                  (setf (aref arr i (- j 1)) (+ (* n i) j)))
          (loop for j downfrom n to 1 do
                (setf (aref arr i (- n j)) (+ (* n i) j)))))))
```

8.2. Результаты работы

Для более удобной читаемости результата, результат функции `matrix-tr-br` будем передовать в `print-matrix`. (Функция взята со страницы курса.)

CL-USER 12 > (print-matrix (matrix-tr-br 1))

#2A((1))

CL-USER 13 > (print-matrix (matrix-tr-br 2))

#2A((2 1)
(3 4))

CL-USER 14 > (print-matrix (matrix-tr-br 3))

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

CL-USER 15 > (print-matrix (matrix-tr-br 4))

#2A((4 3 2 1)
(5 6 7 8)
(12 11 10 9)
(13 14 15 16))

CL-USER 16 > (print-matrix (matrix-tr-br 8))

```
   #2A((8 7 6 5 4 3 2 1)
(9 10 11 12 13 14 15 16)
(24 23 22 21 20 19 18 17)
(25 26 27 28 29 30 31 32)
(40 39 38 37 36 35 34 33)
(41 42 43 44 45 46 47 48)
(56 55 54 53 52 51 50 49)
(57 58 59 60 61 62 63 64))
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Программа работает за $O(n^2)$. Так как мы проходимся по всей матрице.

11. Выводы

В ходе выполнения лабораторной работы я познакомился с массивами. Узнал как с ними удобно работать и какие в нём есть преимущества, например массив знает свой размер. Также получил опыт работы с циклами. Их отличительная черта от циклов, с которыми я привык работать в императивных языках, в том, что они обязательно возвращают значение.