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

Студентка группы М8О-307 МАИ *Довженко Анастасия*, №7 по списку
Контакты: tutkarma@gmail.com
Работа выполнена: 23.03.2019

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

1. Тема работы

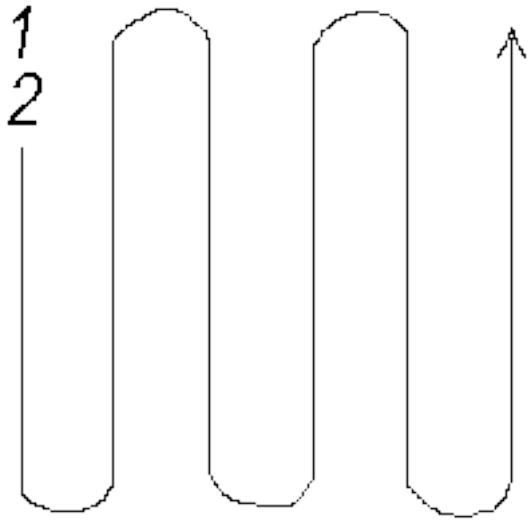
Последовательности, массивы и управляющие конструкции Common Lisp.

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 №3.42)

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.



4. Оборудование студента

Ноутбук Asus UX310U, процессор Intel Core i7-6500U CPU 2.50GHz x 4, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 16.04 LTS, компилятор clisp, текстовый редактор Sublime Text 3.

6. Идея, метод, алгоритм

Заполняем матрицу постолбцово, причем четные столбцы заполняются сверху вниз, а нечетные – снизу вверх. При каждом заполнении увеличиваем локальный счетчик на единицу, что позволяет выполнить условие, при котором элементами матрицы являюся числа от единицы до квадрата исходного числа `n`. На каждом шаге цикла заполняются два столбца – четный и нечетный. Если исходное число `n` – нечетное, то последний нечетный столбец не заполняется, потому что его нет.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun get-matrix (n)
  (let  ((matrix (make-array (list n n)))
         (cnt (ceiling n 2))
         (num 1))

  (dotimes (i cnt)

    (loop for j from 0 upto (1- n)
      do (setf (aref matrix j (* i 2)) num)
         (setf num (1+ num))
    )

    ; условие, чтобы корректно обрабатывались матрицы с нечетным размером
    (when (or (evenp n) (< i (1- cnt)))
      (loop for j from (1- n) downto 0
        do (setf (aref matrix j (1+ (* i 2))) num)
           (setf num (1+ num))
      )
```

```
    )
  )

    matrix)
)


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


(defun matrix-tl-bl (n)
  (get-matrix n)
)


(print-matrix (matrix-tl-bl 1))
(print-matrix (matrix-tl-bl 5))
(print-matrix (matrix-tl-bl 8))
```

8.2. Результаты работы

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

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Я уже встечалась с подобной задачей обхода матрицы в одной из лабораторных на 1 курсе, поэтому алгоритмически она не показалась мне сложной, зато было интересно сравнить реализации двух разных парадигм.

11. Выводы

Я познакомилась с массивами в языке Lisp, а также узнала, как выполнять различные операции над ними, как использовать циклы. Массивы являются основополагающей структурой данных в программировании и часто используются, потому что в них удобно хранить данные. Для языка Lisp существуют целые библиотеки для работы с матрицами, может быть когда-нибудь я напишу свою, всякое может случиться, никогда не знаешь, что тебе пригодится в будущем.