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

Студент группы 8О-307 МАИ *Вельтман Лина*, №7 по списку
Контакты: kluuo@mail.ru
Работа выполнена: 5.04.2020

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

1. Тема работы

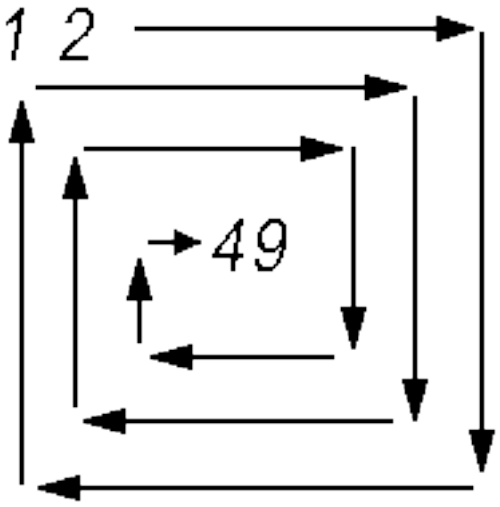
Последовательности, массивы и управляющие конструкции Коммон Лисп.

2. Цель работы

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

3. Задание (вариант №46)

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 спирали.



4. Оборудование ПЭВМ студента

Ноутбук MacBook Pro (13-inch, 2017), процессор 2.3GHz Intel Core i5, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение ЭВМ студента

macOS Catalina 10.15.4, реализация языка SBCL 1.4.16, текстовый редактор Sublime Text 3.

6. Идея, метод, алгоритм

Заполнение матрицы происходит по столбцам, а затем по строкам поочередно. Сначала проходим по столбцам слева-направо (причем при каждом заполнении увеличиваем счетчик на единицу, что позволяет выполнить условие, при котором элементами матрицы являются числа от единицы до квадрата исходного числа n), пока не достигнем границы матрицы. Дойдя до пограничного элемента, мы спускаемся на один элемент и проходим уже по строкам сверху-вниз, заполняя каждый элемент матрицы. Опять достигнув пограничного элемента матрицы, мы начинаем двигаться снова по столбцам, но уже меняем направление: справа-налево. Дойдя до границы, начинаем двигаться по строкам, но уже сверху-вниз. Далее алгоритм повторяется сначала. Стоит отметить, что проверяются не только границы матрицы, но и был ли уже заполнен выбранный элемент, если одно из этих условий выполняется, то переменным, отвечающим за шаги по столбцам и строкам присваиваются новые значения (а именно: шагу по столбцам присваивается значение шага по строкам с минусом, а шагу по строкам присваивается еще неизмененное значение шага по столбцам), а не просто инкрементируются.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun get-matrix (n)
  (if (> 0 n) (return-from get-matrix nil))
  (do ((size (+ 1 (* n n)))
     (mtrx (make-array (list n n) :initial-element -1))
     (step_column 1) (step_row 0) (column 0) (row 0)
     (i 1 (1+ i)))
    ((= i size) mtrx)
```

```
  (setf (aref mtrx row column) i)
  (let ((new_column (+ column step_column)) (new_row (+ row
   step_row)))
    (cond
     ((and (< -1 new_column n)
           (< -1 new_row n)
           (= -1 (aref mtrx new_row new_column)))
      (setf column new_column
            row new_row))
     (t (psetf step_column (- step_row)
               step_row step_column)
        (setf column (+ column step_column)
              row (+ row step_row))))))))

(defun spiral-matrix(n)
  (get-matrix n))

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

8.2. Результаты работы

```
(base) MacBook-Pro-Lina:lab3 linuxoid$ sbcl
This is SBCL 1.4.16, an implementation of ANSI Common Lisp.
More information about SBCL is available at
   <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no
   warranty.
It is mostly in the public domain; some portions are provided
   under
BSD-style licenses.  See the CREDITS and COPYING files in the
distribution for more information.
* (compile-file "./lab3_46.lisp")
; compiling file
   "/Users/linuxoid/Desktop/VUZICH/FP/lab3/lab3_46.lisp" (written
   05 APR 2020 09:11:53 PM):
; compiling (DEFUN GET-MATRIX ...)
```

```
; compiling (DEFUN SPIRAL-MATRIX ...)
; compiling (DEFUN PRINT-MATRIX ...)

; wrote /Users/linuxoid/Desktop/VUZICH/FP/lab3/./lab3_46.fasl
; compilation finished in 0:00:00.024
#P"/Users/linuxoid/Desktop/VUZICH/FP/lab3/lab3_46.fasl"
NIL
NIL
* (load "lab3_46.fasl")
T
* (print-matrix (spiral-matrix 1))

#2A((1))
* (print-matrix (spiral-matrix 5))

#2A((1  2  3  4  5)
    (16 17 18 19 6)
    (15 24 25 20 7)
    (14 23 22 21 8)
    (13 12 11 10 9))
* (print-matrix (spiral-matrix 9))

#2A((1  2  3  4  5  6  7  8  9)
    (32 33 34 35 36 37 38 39 10)
    (31 56 57 58 59 60 61 40 11)
    (30 55 72 73 74 75 62 41 12)
    (29 54 71 80 81 76 63 42 13)
    (28 53 70 79 78 77 64 43 14)
    (27 52 69 68 67 66 65 44 15)
    (26 51 50 49 48 47 46 45 16)
    (25 24 23 22 21 20 19 18 17))
* (print-matrix (spiral-matrix 0))

#2A()
* (print-matrix (spiral-matrix -5))
NIL
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Было интересно вспомнить работу с двумерными матрицами и их обходом, выявлять закономерности, которые помогут упростить алгоритм.

11. Выводы

Во время выполнения данной лабораторной я познакомилась с массивами в языке `Common 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, поэтому важно уметь работать с ними в любом языке программирования.