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

Студент группы 8О-307 МАИ *Ефимов Александр*, №7 по списку
Контакты: aleks.efimov2011@yandex.ru
Работа выполнена: 26.03.2021

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

1. Тема работы

Последовательности, массивы и управляющие конструкции Common Lisp.

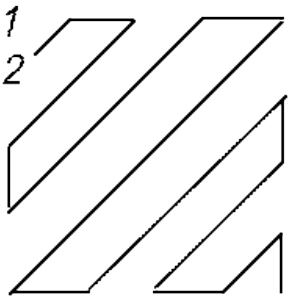
2. Цель работы

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

3. Задание

Вариант: №3.45

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

Процессор Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz, память: 7.6Gi, разрядность системы: 64.

5. Программное обеспечение

ОС Arch Linux, система CLisp.

6. Идея, метод, алгоритм

После создания матрицы, ее можно заполнить пошагово сначала над антидиагональю (диагональ противоположная главной) следующим алгоритмом:

1. Приравнять текущий элемент к текущему шагу;
2. Проверить четность-нечетность текущей побочной диагонали:
   - Если четная, то спуститься до границы или перейти на следующую диагональ, если спускаться некуда;
   - Иначе также, по подниматься.
   - Перейти к шагу 1 если антидиагональ не достигнута

Второй цикл отличается от первого только двумя изменениями:

1. В то время как первый цикл продолжался до достижения антидиагонали, этот цикл продолжается пока общее число выполненных шагов не равно n^2;
2. Переход на следующую диагональ производится с учетом того, что мы на границе.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(defun matrix-1l-2l (n)
  "Creates a zig-zag matrix of size 'n'"
  (let ((matrix (make-array (list n n) :element-type 'integer))
        (col 0)
        (row 0)
        (value 1))
      ; Fills triangle above antidiagonal
      (loop while (and (/= col (1- n)) (/= row (1- n)))
        do (setf (aref matrix col row) value)
           (incf value)
           (if (evenp (+ col row))
               (if (= row 0)
                   (incf col)
                   (setf row (1- row)
                         col (1+ col)))
               (if (= col 0)
```

```
17                      (incf row)
18                      (setf col (1- col)
19                            row (1+ row)))))
20        ; Fills the rest of the matrix
21        (loop while (/= value (1+ (* n n)))
22          do (setf (aref matrix col row) value)
23             (incf value)
24             (if (evenp (+ col row))
25                 (if (= col (1- n))
26                     (incf row)
27                     (setf row (1- row)
28                           col (1+ col)))
29                 (if (= row (1- n))
30                     (incf col)
31                     (setf col (1- col)
32                           row (1+ row)))))
33        ; Returns the resulting matrix
34        matrix))
```

8.2. Результаты работы

```
[1]> (load "zig-zag.lisp")
;; Loading file zig-zag.lisp ...
;; Loaded file zig-zag.lisp
[2]> (matrix-1l-2l 1)
#2A((1))
[3]> (matrix-1l-2l 2)
#2A((1 3) (2 4))
[4]> (matrix-1l-2l 3)
#2A((1 3 4) (2 5 8) (6 7 9))
[5]> (matrix-1l-2l 4)
#2A((1 3 4 10) (2 5 9 11) (6 8 12 15) (7 13 14 16))
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

11. Выводы

- Создание матрицы производиться через функцию *make-array*;

- Цикл *loop* является смесью императивного и функционального программирования;
- И *let*, и *loop* могут иметь в своем теле неограниченное число вызовов.