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

Студент группы 8О-307 МАИ *Бирюков Виктор*, №2 по списку
Контакты: vikvladbir@mail.ru
Работа выполнена: 24.04.2022

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 №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.

```
(defun matrix-1l-2l (n)
  ...)

(matrix-1l-2l 4) =>
#2A((1  3  4 10)
    (2  5  9 11)
    (6  8 12 15)
    (7 13 14 16))
```

4. Оборудование студента

Процессор AMD Ryzen 7 3700U @ 2.3GHz, память: 20Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Windows 10, компилятор SBCL 2.2.2, текстовый редактор Sublime Text 4.

6. Идея, метод, алгоритм

Обход матрицы в таком порядке состоит из последовательных обходов ее побочных диагоналей. При этом направление обхода диагоналей чередуется. Элементы побочных диагоналей обладают следующим свойством — их сумма индексов постоянна и, при нуль-индексации элементов и диагоналей, равна номеру диагонали. Таким образом все элементы побочной диагонали можно однозначно определить, зная номер диагонали и один из индексов самого нижнего или самого верхнего ее элемента.

Функция `fill-side-diag` осуществляет заполнение побочной диагонали с номером m, номером строки крайнего элемента i и в заданном направлении. Диагональ заполняется числами, начиная с idx. Число, следующее за последним в диагонали, возвращается как результат функции. Функция `fill-matrix` заполняет всю матрицу. Номер строки первого элемента – 0, направление обхода – вниз. После заполнения побочной диагонали, номер строки сдвигается в ее конец. Затем происходит перемещение к новой диагонали, при этом необходимо перейти на следующую строку, если мы находимся в первом или последнем столбце. В иных случаях перемещение происходит исключительно за счет изменения m. Направление обхода изменяется на противоположное.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
; 3.45

(defun fill-side-diag (matr j i m n idx up)
    (if (< j (min (+ m 1) (- (* 2 n) m 1)))
        (progn
            (setf (aref matr i (- m i)) idx)
            (fill-side-diag matr (+ j 1) (+ i (if up -1 1)) m n (+ idx 1) up))
        idx))

(defun fill-matrix (matr i m n idx up)
    (if (< m (- (* 2 n) 1))
        (let
            ((next-i
                (let
                    ((next-i (+ i (* (min m (- (* 2 n) m 2)) (if up -1 1)))))
                    (+ next-i
                       (if (and (/= next-i (- n 1))
```

```
                                (or (= (- m next-i) 0)
                                    (= (- m next-i) (- n 1))))
                            1 0))))
            (next-idx (fill-side-diag matr 0 i m n idx up)))
            (fill-matrix matr next-i (+ m 1) n next-idx (not up)))
        matr))

(defun matrix-1l-2l (n)
    (fill-matrix (make-array (list n n)) 0 0 n 1 NIL))

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

8.2. Результаты работы

```
* (print-matrix (matrix-1l-2l 4))

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

* (print-matrix (matrix-1l-2l 2))

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

* (print-matrix (matrix-1l-2l 5))

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

* (print-matrix (matrix-1l-2l 3))

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

* (print-matrix (matrix-1l-2l 8))
```

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

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Вспомогательные функции описывают линейно-итеративные процессы. Сложность алгоритма — $O(n^2)$ по времени, $O(n^2)$ по памяти.

11. Выводы

В ходе выполнения лабораторной работы я познакомился со структурой данных массив, при помощи которого можно представлять матрицы произвольных размерностей. Непосредственная работа с элементами матрицы делает взаимодействие с ними похожим на взаимодействие в императивных языках.