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

Студент группы 8О-307Б-18 МАИ *Тояков Артем*, №22 по списку
Контакты: temathesuper@mail.ru
Работа выполнена: 12.04.2021

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

Изучить Последовательности, массивы и управляющие конструкции Коммон Лисп.

## 3. Задание (вариант № 3.17)

Запрограммировать на языке Коммон Лисп функцию, принимающую три аргумента:

- A - двумерный массив, представляющий действительную матрицу размера m×n,
- u - вектор действительных чисел длины n,
- i - номер строки, $0 \leqslant i \leqslant m$.

Функция должна возвращать новую матрицу размера (m+1)×n, полученную вставкой после строки с номером i новой строки с элементами из u. i=0 означает вставку перед первой строкой.

Исходный массив A должен оставаться неизменным.

## 4. Оборудование студента

Процессор: Intel(R) Core(TM) i7-8565U CPU @ 1.80GHz, память: 3,8 Gb, разрядность системы: 64.

## 5. Программное обеспечение

UBUNTU 18.04.5 LTS, компилятор sbcl

# 6. Идея, метод, алгоритм

Метод состоит из двух итераций. Идея в том, чтобы для начала вставить u-вектор на место i+1 строки матрицы b. Затем же в полученную матрицу на место j+1 столбца вставить v-вектор. Мне предстоит разбить матрицу на составные части и по новому собрать ее. В программе две основные функции:

- (line-extended-matrix a k v) - В данной функции происходит создание матрицы b и копирование элементов матрицы до элемента a[k][...]. Затем же вставка вектора строки на позицию b[k][...] и копирование остальных элементов, если оно необходимо.
- (extended-matrix a u i) - Функция-вызов для функции line-extended-matrix, где учитываются ограничения, наложенные на индекс i.

# 7. Сценарий выполнения работы

- Анализ возможных реализаций поставленной задачи на common Lisp
- Изучение синтаксиса и основных функций работы со списками common Lisp
- Реализация поставленной задачи на common Lisp

# 8. Распечатка программы и её результаты

## 8.1. Исходный код

```
(defun line-extended-matrix (a k v)
    (let ((m (array-dimension a 0))
        (n (array-dimension a 1)))
        (let ((b (make-array (list (+ 1 m) n)))) ; b = Matrix(* (1 + m) n)
            (dotimes (i k)
                (dotimes (j n)
                    (setf (aref b i j) (aref a i j)))) ; b[i][j] = a[i][j]
            (loop
                :for j :below n
                :do (setf (aref b k j) (aref v j))) ; b[k][j] = v[j], j : 0 to n
            (dotimes (j n)
                (loop :for i :from k :to (- m 1) :do
                    (setf (aref b (+ 1 i) j) (aref a i j)))) ; b[i + 1][j] = a[i][j]
        b)
    )
)

(defun extended-matrix (a u i)
    (array-dimension a 0)
```

```
    (array-dimension a 1)
    (if (and (<= i (array-dimension a 0)) (>= i 0)) (line-extended-matrix a i u))
)
```

### 8.2. Результаты работы

Файл с тестовыми переменными прикреплён к работе.

```
* (extended-matrix m1 u1 0)

#2A((1 1 1) (0 0 0) (0 0 0) (0 0 0))
* (extended-matrix m2 u2 1)

#2A((0 0 0 0) (1 1 1 1) (0 0 0 0))
* (extended-matrix m3 u3 2)

#2A((0 0 0 0) (0 0 0 0) (1 1 1 1) (0 0 0 0))
* (extended-matrix m4 u4 0)

#2A((1) (0))
* (extended-matrix m4 u4 1)

#2A((0) (1))
* (extended-matrix m4 u4 2)

NIL
* (extended-matrix m5 u5 0)

#2A((1 1) (0 0) (0 0) (0 0) (0 0))
* (extended-matrix m5 u5 4)

#2A((0 0) (0 0) (0 0) (0 0) (1 1))
```

## 9. Дневник отладки

| Дата | Событие | Действие по исправлению | Примечание |
|---|---|---|---|
| | | | |

## 10. Замечания автора по существу работы

Довольно долго пришлось разбираться с управляющими конструкциями в Common Lisp.

# 11. Выводы

В ходе данной работы мне удалось познакомиться со встроенными функциями/инструментами, а также управляющими конструкциями коммон лисп с помощью которых мне удалось реализовать классический обход по матрице, используя циклы. Это поможет мне легче понимать, как работает язык и облегчит будущую работу с массивами и матрицами.