# Отчет по лабораторной работе № 4
# по курсу «Функциональное программирование»

Студент группы 8О-307 МАИ *Бирюков Виктор*, №2 по списку
Контакты: vikvladbir@mail.ru
Работа выполнена: 10.05.2022

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

## 1. Тема работы

Знаки и строки.

## 2. Цель работы

Научиться работать с литерами (знаками) и строками при помощи функций обработки строк и общих функций работы с последовательностями.

## 3. Задание (вариант №4.38)

Запрограммировать на языке Коммон Лисп функцию, принимающую один аргумент — натуральное число $n$, $n < 1000000$. Функция должна вернуть предложение, которое выражает это число русскими словами.

## 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

Структура количественных числительных в русском языке такова, что цифры числа можно обрабатывать группами по три, при необходимости вставляя после такой группы слова «тысяча», «миллион» и т.д.

В пределах группы каждый разряд представляется одним словом, за исключением чисел $11 - 19$, которые обозначаются одним словом на два разряда.

Поправка на падеж нужна в двух случаях. Существует три варианта написания трехразрядных слов — «тысяча» для 1, «тысячи» для $2-4$, «тысяч» для $0$, $5-9$, а также $11-19$. Миллионы и более аналогично. Также для тысяч менятся написание: «один» — на «одна», «два» — на «две».

Все необходимые константы записаны в шесть глобальных массивов. Обработку трехзначных групп осуществляет функция `three-digits-sentence`. Обработку всего числа — функция `number-sentence-iter`, которая описывает линейно-итеративный процесс. Таким образом, сложность алгоритма по времени — $O(\log_{1000} n)$

# 7. Сценарий выполнения работы

# 8. Распечатка программы и её результаты

## 8.1. Исходный код

```
; 4.38

(defvar *digits* #("один" "два" "три" "четыре" "пять" "шесть" "семь" "восемь" "девять"))
(defvar *digits-for-thousand* #("одна" "две"))
(defvar *two-digits* #("одиннадцать" "двенадцать" "тринадцать" "четырнадцать"
    "пятнадцать" "шестнадцать" "семнадцать" "восемнадцать" "девятнадцать"))
(defvar *tens* #("десять" "двадцать" "тридцать" "сорок" "пятьдесят" "шестьдесят" "семьдесять"
    "восемьдесят" "девяносто"))
(defvar *hundreds* #("сто" "двести" "триста" "четыреста" "пятьсот" "шестьсот" "семьсот"
    "восемьсот" "девятьсот"))
(defvar *case* #(#("тысяча" "тысячи" "тысяч") #("миллион" "миллиона" "миллионов")))

(defun three-digits-sentence (num pos)
    (let ((res NIL))
        (if (= num 0)
            res
            (let
                ((digit (mod num 10))
                (ten (mod (floor num 10) 10))
                (hundred (mod (floor num 100) 10)))

                (if (/= hundred 0) ; 100 - 900
                    (setq res (concatenate 'string
                                res
                                (svref *hundreds* (1- hundred))
                                " ")))

                (if (and (= ten 1) (/= digit 0)) ; 11 - 19
                    (setq res (concatenate 'string
                                res
                                (svref *two-digits* (1- digit))
                                " "))
                    (progn
```

```
                    (if (/= ten 0) ; 10 - 90
                        (setq res (concatenate 'string
                                    res
                                    (svref *tens* (1- ten))
                                    " ")))
                    (cond
                        ((and (/= digit 0)
                              (or (/= pos 1) (> digit 2))) ; 1 - 9
                            (setq res (concatenate 'string
                                        res
                                        (svref *digits* (1- digit))
                                        " ")))
                        ((/= digit 0) ; 1 - 2 thousand
                            (setq res (concatenate 'string
                                        res
                                        (svref *digits-for-thousand*
                                               (1- digit))
                                        " "))))))

                (if (> pos 0)
                    (cond
                        ((or (= digit 0) (= ten 1) (> digit 4))
                            (setq res (concatenate 'string
                                        res
                                        (svref (svref *case* (1- pos)) 2)
                                        " ")))
                        ((= digit 1)
                            (setq res (concatenate 'string
                                        res
                                        (svref (svref *case* (1- pos)) 0)
                                        " ")))
                        (T
                            (setq res (concatenate 'string
                                        res
                                        (svref (svref *case* (1- pos)) 1)
                                        " "))))))
                res))))

(defun number-sentence-iter (num pos res)
    (if (= num 0)
        res
        (number-sentence-iter
            (floor num 1000)
            (1+ pos)
            (concatenate 'string
                (three-digits-sentence (mod num 1000) pos)
                res))))

(defun number-sentence (num)
    (string-right-trim " " (number-sentence-iter num 0 NIL)))
```

### 8.2. Результаты работы

```
* (number-sentence 3967)
"три тысячи девятьсот шестьдесят семь"

* (number-sentence 1004012)
"один миллион четыре тысячи двенадцать"

* (number-sentence 925101503)
"девятьсот двадцать пять миллионов сто одна тысяча пятьсот три"

* (number-sentence 143054637)
"сто сорок три миллиона пятьдесят четыре тысячи шестьсот тридцать семь"

* (number-sentence 999999999)
"девятьсот девяносто девять миллионов девятьсот девяносто девять тысяч девятьсот
    девяносто девять"
```

## 9. Дневник отладки

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

## 10. Замечания автора по существу работы

Так как кроме тысяч исключений больше нет, величина обрабатываемых чисел может быть увеличена путем расширения массива `*case*`.

## 11. Выводы

В ходе выполнения лабораторной работы я познакомился со строками и последовательностями в языке Коммон Лисп, а также с функциями для их обработки.