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

Студент группы 8О-306 Кузьмичев Александр, № по списку 12.

Контакты: alcuzmichev@yandex.ru  
Работа выполнена: 30.04.2021  
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 № 4.37)

Запрограммировать на языке Коммон Лисп функцию find-word, принимающую два аргумента[:](http://lisp.ystok.ru/fp/lisp/string_24.html#text)

* word - строка, представляющая [слово](http://lisp.ystok.ru/fp/lisp/string_24.html#word),
* source - [текст](http://lisp.ystok.ru/fp/lisp/string_24.html#text).

Если слово найдено, функция должна возвращать два значения с помощью [values](http://lisp.ystok.ru/fp/lisp/1_38.html" \l "VALUES" \o "Множественное значение и VALUES):

1. индекс начала данного слова в предложении,
2. номер первого предложения в тексте, в которое входит слово (нумерация с 0).

Если слово не найдено, функция должна вернуть NIL.

### 4. Оборудование студента

Оборудование ПЭВМ студента.

### 5. Программное обеспечение

Программное обеспечение ЭВМ студента.

### 6. Идея, метод, алгоритм

Основная идея данной реализации заключается в последовательном парсинге текста. Счетчик позиции увеличивается каждый раз, когда программа встречает пробел, кавычки, запятую и т.д.

### 7. Сценарий выполнения работы

### 8. Распечатка программы и её результаты

**Программа**

(defun russian-upper-case-p (char)

(position char "АБВГДЕЁЖЗИЙКЛМНОПРСТУФХЦЧШЩЪЫЬЭЮЯ"))

(defun russian-char-downcase (char)

(let ((i (russian-upper-case-p char)))

(if i

(char "абвгдеёжзийклмнопрстуфхцчшщъыьэюя" i)

(char-downcase char))))

(defun russian-string-downcase (string)

(map 'string #'russian-char-downcase string))

(defun parsing\_str (word str)

(let ((ind (search word str)))

(progn

(if ind

(

cond

((= (length str) (+ ind (length word))) ind)

((string= (char str (+ ind (length word))) #\Space) ind)

((string= (char str (+ ind (length word))) #\Tab) ind)

((string= (char str (+ ind (length word))) #\Newline) ind)

((string= (char str (+ ind (length word))) ",") ind)

((string= (char str (+ ind (length word))) ":") ind)

((string= (char str (+ ind (length word))) ";") ind)

((string= (char str (+ ind (length word))) "-") ind)

((string= (char str (+ ind (length word))) "(") ind)

(t (+ ind (length word)))

)

nil

)

)

)

)

(defun double\_find (word index sentence)

(if index

(let

((substr (subseq sentence index) ) (prev\_index -1) (is-null (lambda (obj)(if (not obj) (return-from double\_find nil)))))

(progn

(loop while (and (and (/= prev\_index index) prev\_index) (> (length substr) (length word)))

do (progn

(setq prev\_index (parsing\_str word substr))

(funcall is-null prev\_index)

(setq substr (subseq substr prev\_index))

(setq prev\_index (+ index prev\_index))

)

)

(if index (return-from double\_find index) nil)

)

)

nil

)

)

(defun finding-the-word (word text)

(if (and (/= (length word) 0) word)

(let ((word\_index nil) (sentences\_amount -1) (not-null (lambda (obj)(not (null obj)))))

(progn

(loop for sentence in text

until (funcall not-null word\_index)

do (progn

(incf sentences\_amount)

(setf word\_index (parsing\_str (russian-string-downcase word) (string-right-trim ",.;:?!" (russian-string-downcase sentence))))

(setf word\_index (double\_find (russian-string-downcase word) word\_index (string-right-trim ",.;:?!" (russian-string-downcase sentence))))

)

)

(if (funcall not-null word\_index) (list word\_index sentences\_amount) nil)

)

)

nil

)

)

(defun find-the-word (word text)

(if (finding-the-word word text) (values (car (finding-the-word word text)) (cadr (finding-the-word word text))) nil))

**Результаты**

(print(find-the-word "case" '("Case number one." "Number two case"))) =>

0

0

(print(find-the-word "case" '("number one caSe." "number two case"))) =>

11

0

(print(find-the-word "case" '("Noname test." "Second noname test."))) =>

NIL

### 9. Дневник отладки

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Дата, время | Событие | Действие по исправлению | Примечание |
| 1 |  |  |  |  |

### 10. Замечания автора по существу работы

Выполняя четвертую лабораторную работу по функциональному программированию, я понял, что работать со знаками и строками в языке Common Lisp намного сложнее, чем в языках, с которыми я работал до этого.

### 11. Выводы

В четвертой лабораторной работе я поработал с функциями обработки строк. Работа со строками в языке Common Lisp показалась мне не слишком удобной. Хотя, возможно, за время выполнения одной ЛР я просто не уловил всех тонкостей.