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

Студент группы 8О-307 МАИ *Вельтман Лина*, №7 по списку
Контакты: kluuo@mail.ru
Работа выполнена: 25.04.2020

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 №37)

Запрограммировать на языке Коммон Лисп функцию `find-word`, принимающую два аргумента: `word` - строка, представляющая слово, `source` - текст. Если слово найдено, функция должна возвращать два значения с помощью `values`: индекс начала данного слова в предложении, номер первого предложения в тексте, в которое входит слово (нумерация с 0). Если слово не найдено, функция должна вернуть NIL.

4. Оборудование ПЭВМ студента

Ноутбук MacBook Pro (13-inch, 2017), процессор 2.3GHz Intel Core i5, память: 8Gb, разрядность системы: 64.

5. Программное обеспечение ЭВМ студента

macOS Catalina 10.15.4, реализация языка SBCL 1.4.16, текстовый редактор Sublime Text 3.

6. Идея, метод, алгоритм

Точкой отсчета для меня была встроенная функция `search`. Она позволяет найти заданную подпоследовательность в области поиска. Если вхождение было, то функция возвращает позицию первого вхождения в последовательности, в которой происходил поиск подпоследовательности, иначе возвращается `nil`. Отталкиваясь от этого, я решила последовательно проходить по строкам текста `text` (задан как список, в котором хранятся строки - предложения текста), инкрементируя переменную отвечающую за количество предложений в тексте. Далее для поиска индекса начала слова я написала функцию `parsing_str`, в которой вызывается функция `search` для поиска заданного слова `word` в рассматриваемом предложении `sentence`, затем проверяется, что стоит после конца слова (если было найдено вхождение). Если это пробел, знак табуляции, новая линия или знаки пунктуации, то найденное слово верное и возвращается индекс его начала, если нет, то сдвигаемся от найденной позиции на количество символов, равное длине слова. Затем вызываем еще одну функцию `double_find`, которая проверяет позицию начала вхождения, ведь найденное слово может быть всего лишь подстрокой какого-то слова, а это нам не подходит. Цикл прохода по всем предложениям текста в функции `finding-the-word` работает до тех пор, пока функция `search` не вернет позицию вхождения. Для проверки того, что вернула функция `search` было решено использовать анонимную функцию, реализуемую с помощью лямбда-выражения, в которую в качестве аргумента передавалось возвращаемое значение функции `search`. Если объект(позиция вхождения) существует, то есть не равно `null`, то возвращается `true`, иначе `false`. Цикл прохода по тексту будет работать до тех пор, пока не закончатся предложения (то есть до конца списка) или, если возращаемое значение анонимной функции было `true`, то есть мы нашли вхождение слова в каком-либо предложении. Далее происходит проверка на существование позиции вхождения, если она есть, то создается список, в котором хранятся позиция вхождения слова в предложение и номер предложения, в котором нашлось слово, иначе возвращается `nil`, то есть вхождения заданного слова `word` не было вовсе.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
(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))))          ; латиница


(defun russian-string-downcase (string)
  ;; Преобразовать и латинские, и русские буквы строки в строчные
  (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))
```

8.2. Результаты работы

```
MacBook-Pro-Lina:lab4 linuxoid$ sbcl
This is SBCL 1.4.16, an implementation of ANSI Common Lisp.
More information about SBCL is available at
   <http://www.sbcl.org/>.

SBCL is free software, provided as is, with absolutely no
   warranty.
It is mostly in the public domain; some portions are provided
   under
BSD-style licenses.  See the CREDITS and COPYING files in the
distribution for more information.
* (compile-file "./lab4_37.lisp")
; compiling file
   "/Users/linuxoid/Desktop/VUZICH/FP/lab4/lab4_37.lisp" (written
   10 MAY 2020 07:01:00 PM):
; compiling (DEFUN RUSSIAN-UPPER-CASE-P ...)
; compiling (DEFUN RUSSIAN-CHAR-DOWNCASE ...)
; compiling (DEFUN RUSSIAN-STRING-DOWNCASE ...)
; compiling (DEFUN PARSING_STR ...)
; compiling (DEFUN DOUBLE_FIND ...)
; compiling (DEFUN FINDING-THE-WORD ...)
; compiling (DEFUN FIND-THE-WORD ...)

; wrote /Users/linuxoid/Desktop/VUZICH/FP/lab4/./lab4_37.fasl
; compilation finished in 0:00:00.039
#P"/Users/linuxoid/Desktop/VUZICH/FP/lab4/lab4_37.fasl"
NIL
NIL
* (load "lab4_37.fasl")
T
* (find-the-word знать"" Неприятности'(" случаются." Главное" - это
   знать, что тебе всегда придут на помощь твои друзья!"))
14
1
* (find-the-word б"" Когда'(" бы жизнь домашним кругом" я" б
   ограничить захотел."))
2
```

```
1
* (find-the-word "house" '("I have a big field." "I've
   constructed a wonderful house on this field." "I'm looking
   forward for you to see it!"))
29
1
* (find-the-word "mouse" '("I have a big field." "I've
   constructed a wonderful house on this field." "I'm looking
   forward for you to see it!"))
NIL
* (find-the-word "" '())
NIL
* (find-the-word "Girl" '("You're a pretty girl!" "Girl, help me
   with this task, please." "I cannot find my book..."))
16
0
* (find-the-word "fOrCe" '("Hello there!" "General Kenobi!" "MaY
   tHe FoRcE bE wItH yoU!" "My YoUng PadAwaN!" "ThrouGh the fOrCe
   wE gaIn pOweRrRrr!"))
8
2
* (find-the-word "9nine9" '("8888! iujo" "9 8 9 9" "9nine 9"
   "!9nine! and twenty four" "and that's 9nine9"))
11
4
* (find-the-word Захотел"" Я'(" бы очень хотела поступить подругому-."
   Но" он решительно захотел напомнить мне, что захотел для меня иной
   участи. "))
17
1
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

В реализации мне помогла встроенная функции поиска подпоследовательностей `search` и `lambda` - определяет анонимную функцию, она обычно используется, когда при определении новая функции не именуется. Это так же делает замысел программиста более очевидным, за счет того, что функция находится там же, где она используется.

11. Выводы

Я получила опыт работы со строками при помощи общих функций для строк и последовательностей. Также для выполнения данной лабораторной работы мне помогли знания, полученные в предыдущих работах.