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

Студент группы 8О-306 Гаврилов Максим, № по списку 7.

Контакты: cobraj@yandex.ru
Работа выполнена: 03.05.2022
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 № 4.42)

Запрограммировать на языке Коммон Лисп функцию, принимающую два аргумента:

- ch - произвольный знак,
- text - текст.

Функция должна возвращать список слов, в которых либо первая буква, либо последняя совпадает с ch. Список не должен содержать повторения. Сравнение как латинских букв, так и русских должно быть регистро-независимым.

```
(collect-words-with-char #\T
 '("Чего только не придумает солдатская смекалка."
   "Вот очередной пример."))
 => ("только" "придумает" "Вот")
```

4. Оборудование студента

процессор AMD FX(tm)-6300 Six-Core Processor 3.50 GHz, память 16ГБ, 64-разрядная система.

5. Программное обеспечение

ОС Windows 10, программа portacle, версия slime 2.24

6. Идея, метод, алгоритм
Итеративный процесс с использованием цикла

7. Сценарий выполнения работы

1. Изучить функции lisp для работы со строками.
2. Определить, какие функции необходимы для выполнения задания.
3. Если необходимо, написать вспомогательные функции.

8. Распечатка программы и её результаты

Программа

```
;; Функции из лекций
(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 whitespace-char-p (char)
  (member char '(#\Space #\Tab #\Newline)))

(defun word-list (string)
  (loop with len = (length string)
    for left = 0 then (1+ right)
    for right = (or (position-if #'whitespace-char-p string :start left) len)
    unless (= right left)   ; исключить пустые слова
      collect (subseq string left right)
      while (< right len)))

;; мои функции
(defun remove-punctuation (list-text)
  ;;удаление пунктуации из списка слов текста, разбитого по предложениям
  (loop
    for sentence in list-text
    collect (loop
      for word in sentence
        collect (string (string-right-trim ",.;:?!" word)))))

(defun str-eql (lhs rhs)
  ;; регистронезависимое сравнение строк
  (equal (russian-string-downcase lhs) (russian-string-downcase rhs)))

(defun rm (elm list)
  ;;удаление строки elm из списка
  (cond
    ((null list ) nil)
    ((str-eql elm (first list)) (rm elm (rest list)))
    (t (cons (first list) (rm elm (rest list))))))

 (defun rm-rpt (list)
  ;;удаление поторяющихся элементов из списка
  (if  (null list)
    nil
    (cons (first list) (rm-rpt (rm (first list) (rest list))))))

(defun unite (list-text)
  ;;объединение списка списков в один список
  (if (null list-text)
    nil
    (append (first list-text) (unite (rest list-text)))))
```

```
(defun collect-words-with-char (ch text)
  ;;перевод текста в список неповторяющихся слов в нижнем регистре
  ;;затем для каждого слова из списка проверяется соответствие первой или последней буквы
заданному ch
  (loop
    for word in (rm-rpt (unite  (remove-punctuation (mapcar #'(lambda (x) (word-list x)) text)) ))
    when (or (eql (russian-char-downcase (char word 0)) (russian-char-downcase ch))
             (eql  (russian-char-downcase (char (reverse word) 0)) (russian-char-downcase ch)))
    collect word))
```

Результаты

```
; SLIME 2.24
CL-USER> (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 whitespace-char-p (char)
  (member char '(#\Space #\Tab #\Newline)))

(defun word-list (string)
  (loop with len = (length string)
    for left = 0 then (1+ right)
    for right = (or (position-if #'whitespace-char-p string :start left) len)
    unless (= right left)   ; исключить пустые слова
      collect (subseq string left right)
      while (< right len)))


(defun remove-punctuation (list-text)
  ;;удаление пунктуации из списка слов текста, разбитого по предложениям
  ;;и приведение его слов к нижнему регистру
  (loop
    for sentence in list-text
    collect (loop
      for word in sentence
        collect (string (string-right-trim ",.;:?!" word)))))

(defun str-eql (lhs rhs)
  (equal (russian-string-downcase lhs) (russian-string-downcase rhs)))

(defun rm (elm list)
  ;;удаление элемента elm из списка
  (cond
    ((null list ) nil)
    ((str-eql elm (first list)) (rm elm (rest list)))
    (t (cons (first list) (rm elm (rest list))))))
```

```
 (defun rm-rpt (list)
  ;;удаление поторяющихся элементов из списка
  (if  (null list)
    nil
    (cons (first list) (rm-rpt (rm (first list) (rest list))))))

(defun unite (list-text)
  ;;объединение списка списков в один список
  (if (null list-text)
    nil
    (append (first list-text) (unite (rest list-text)))))


(defun collect-words-with-char (ch text)
  ;;перевод текста в список неповторяющихся слов в нижнем регистре
  ;;затем для каждого слова из списка проверяется соответствие первой или последней буквы
заданному ch
  (loop
    for word in (rm-rpt (unite  (remove-punctuation (mapcar #'(lambda (x) (word-list x)) text)) ))
    when (or (eql (russian-char-downcase (char word 0)) (russian-char-downcase ch))
             (eql  (russian-char-downcase (char (reverse word) 0)) (russian-char-downcase ch)))
    collect word))
COLLECT-WORDS-WITH-CHAR
CL-USER> (collect-words-with-char #\T
 '("Чего только не придумает солдатская смекалка."
   "Вот очередной пример."))
   ;;пример из задания
("только" "придумает" "Вот")
CL-USER> (collect-words-with-char #\T
 '("Чего только не придумает солдатская смекалка."
   "Вот Только очередной Не пример."))
   ;;убедимся, что повторы не меняют результат
("только" "придумает" "Вот")
CL-USER> (collect-words-with-char #\p
 '("Чего только не придумает солдатская смекалка."
   "Вот Только очередной Не пример."))
   ;;пример с другой буквой, убедимся, что
   ;;знаки пунктуации не мешают нахождению слов
("пример")
CL-USER> (collect-words-with-char #\y
  (list "We are happy." "What about you?" "Hehe rY."))
   ;;пример с латиницей. Убедимся, что и с ней
   ;;регистр не влияет на поиск
("happy" "you" "rY")
CL-USER> (collect-words-with-char #\y
  (list "We are happy." "What about you?" "Hehe not like You!"))
   ;;повторения в латинице. также не влияют на ответ.
("happy" "you")
CL-USER>
```

9. Дневник отладки

№	Дата, время	Событие	Действие по исправлению	Примечание
1				

10. Замечания автора по существу работы

В работе я использовал функции, определенные в курсе лекций по функциональному программированию, а именно:
russian-upper-case-p,
russian-char-downcase,
russian-string-downcase,
whitespace-char-p,
word-list.

Также я самостоятельно разработал функции по работе со списками, необходимые для промежуточной обработки получившегося после применения word-list списка слов.
stq-eql (lhs rhs) – регистронезависимое сравнение строк.
rm (elm list) – удаление строки elm из списка.
rm-rpt (list) – удаление повторов из списка.
remove-punctuation (list-text) – удаление знаков пунктуации из текста, обработанного функцией word-list
unite (list-text) – объединение списка списков в один список

Вначале с помощью mapcar функция word-list применяется к каждому предложению текста, затем из получившегося списка списков слов функцией remove-punctuation удаляются знаки пунктуации. Потом выполняется объединение списков слов каждого предложения в один список всех слов текста, который обрабатывается функцией rm-rpt для исключения повторений. После этого по списку выполняется проход циклом loop со сбором тех слов, первый или последний символ которых равен искомому символу в нижнем регистре.

Изначально функция remove-punctuation также приводила все слова к нижнему регистру, но, так как я позже заметил, что в примере возвращаемые функцией слова имеют такую же капитализацию, в какой они были в тексте, я убрал приведение слов в нижний регистр и стал использовать функцию регистронезависимого сравнения str-eql. Таки образом, слова с разной капитализацией все еще считаются одним и тем же словом, но при этом в ответ слово попадет в таком виде, в каком оно встретилось в тесте впервые, что хорошо видно во втором примере с латиницей. Мне лично больше нравится первый мой вариант, где все слова приводились к нижнему регистру, так как там не нарушается универсальность функции rm-rpt, которую после модификаций можно применять только к спискам строк, а прежде можно было использовать с любыми списками. Я полагаю, также можно было просто считать слова с разной капитализацией разными словами, что иногда может иметь смысл, но чаще выглядит странно, ведь обычно слово, начинающееся с заглавной буквы, просто стоит в начале предложения и означает то же самое, что и слово, начинающееся со строчной.

11. Выводы

В ходе выполнения этой лабораторной работы я получил опыт написания на языке Коммон Лисп функций, осуществляющих обработку текста в частности и списков вообще. Как-то получилось, что я даже больше работал со списками, чем со строками, ведь разбиение строк на списки слов моя функция осуществляет первым делом. Мне кажется очень полезным то, что в ходе выполнения этой работы я написал много небольших рекурсивных функций, тем самым закрепив навыки, полученные в ходе выполнения первых лабораторных работ.