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

Студент группы М8О-307-19 МАИ *Инютин Максим Андреевич*, №10 по списку
Контакты: mainyutin@gmail.com
Работа выполнена: 20.05.2022

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 № 4.44)

Запрограммировать на языке Коммон Лисп функцию с двумя параметрами:

- char-bag — список знаков,
- text — текст.

Функция должна вернуть преобразованный текст, во всех предложениях которого все слова «обтёсаны», т.е. в них с обоих концов, удалены знаки, упомянутые в char-bag.

4. Оборудование студента

Процессор Intel Core i7-9750H (12) @ 4.5GHz, память: 32 Gb, разрядность системы: 64.

5. Программное обеспечение

ОС Ubuntu 20.04.4 LTS, комилятор GNU CLISP 2.49.92, текстовый редактор Atom 1.58.0

6. Идея, метод, алгоритм

Текстом называется список предложений. Предложение представляется строкой и состоит из слов, разделённых пробелами, знаками горизонтальной табуляции или перевода строки. Возможно больше одного пробела между двумя словами.

Функция text-trim отделяет голову текста, обрабатывает предложение и добавляет его в результирующий список. Так результат будет в обратном порядке, поэтому нужно делать реверс. Для обработки предложений я разбиваю строку на слова и обтёсываю их. Результат — конкатенация обтёсанных слов и тех же разделителей, что и в исходной строке.

Для обтёсывания я использую встроенную функцию string-trim. Её сложноть неизвестна, но я бы реализовал её так: сохранял бы все символы в хэш-таблицу, удалял символы с конца, пока они в таблице, сделал бы реверс строки и повторил процедуру, снова реверс строки. Если считать сложность обращения к хэш-таблице $O(1)$, то функция имела бы сложность $O(|s|)$ для строки s. Буду считать, что она действительно такая. Реверс списка тоже имеет линейную сложность, поэтому асимптотика алгоритма $O(n + \sum_{i=1}^{n} |s_i|)$, где n — количество предложений в тексте, а $s_i - i$ предложение.

7. Сценарий выполнения работы

8. Распечатка программы и её результаты

8.1. Исходный код

```
; 4.44

; Is character a delimiter?
(defun delimiterp (c)
       (member c '(#\Space #\Tab #\Newline)))

; Trims a word
(defun process-word (bag str)
       (if (null str) str (string-trim bag str)))

; Get substring with indexes [l, r) from str
; Returns empty string if l >= r
(defun substr (str l r)
       (if (>= l r) "" (subseq str l r)))

; Trims all words in a string
(defun process-string (bag str)
       ; Add one space to end of str to do all trim in one loop
```

```
        (let* ((s (concatenate 'string str " ")) (left -1) (n
    (length s)) (res ""))
        (loop for right from 0 to (- n 1)
                do (if (delimiterp (char s right)) (progn
                        (setq res (concatenate 'string res
    (process-word bag (substr s (+ left 1) right))))
                        (setq left right)
                        (setq res (concatenate 'string res
    (coerce (list (char s right)) 'string))))))
        (subseq res 0 (- (length res) 1))))

(defun recursive-text-trim (bag lst res)
        (if (null lst) res
        (recursive-text-trim bag (rest lst) (cons
    (process-string bag (first lst)) res))))

(defun text-trim (bag lst)
        (let ((lst-copy lst) (res '()))
        (reverse (recursive-text-trim bag lst-copy res))))
```

8.2. Результаты работы

```
[1]> (load "lab4.lisp")
;; Loading file lab4.lisp ...
;; Loaded file lab4.lisp
#P"/home/engineerxl/Study/6
   term/functional-programming/lab4/lab4.lisp"
[2]> (text-trim '(#\, #\. #\Б #\б)
                '("Блажен, кто смолоду был молод,"
                  "Блажен, кто вовремя созрел."))
("лажен кто смолоду ыл молод" "лажен кто вовремя созрел")
[3]> (text-trim '(#\, #\. #\Б #\б)
                '("Блажен,
                кто смолоду был молод,"
                "Блажен,
                кто вовремя созрел."))
("лажен
                кто смолоду ыл молод"
 "лажен
                кто вовремя созрел")
[5]> (setq bag '(#\а #\б #\p))
(#\CYRILLIC_SMALL_LETTER_A #\CYRILLIC_SMALL_LETTER_BE
   #\CYRILLIC_SMALL_LETTER_ER)
[6]> (setq a '("абракадабра" "аавабрабрав" "" "тут много
   пробелов" "абра"))
("абракадабра" "аавабрабрав" "" "тут много пробелов" "абра")
[7]> (text-trim bag a)
("кад" "вабрабрав" "" "тут много пробелов" "")
[8]> a
("абракадабра" "аавабрабрав" "" "тут много пробелов" "абра")
```

9. Дневник отладки

Дата	Событие	Действие по исправлению	Примечание

10. Замечания автора по существу работы

Сложность полученного решения $O(n + \sum_{i=1}^{n} |s_i|)$. Добиться лучшей асимптотики нельзя, так как сами данные имеют такую же размерность. Можно попробовать уменьшить константу за счёт написания своего string-trim и использования другой структуры данных для хранения набора удаляемых символов.

11. Выводы

Я познакомился со строками и литерами в Common Lisp. Очень приятно использовать высокоуровневые функции, например subseq, встроенные в язык. Чем-то напоминает Python, в котором так же легко работать со строками.