Міністерство освіти і науки України
Національний технічний університет України «КПІ ім. Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

Мультипарадигменне програмування

ЗВІТ

з лабораторної роботи № 2
Варіант №1

Виконав студент групи ІП-71, Амброс Всеволод Володимирович

(№ групи, прізвище, ім’я, по батькові)

Прийняв Очеретяний Олександр Костянтинович

(посада, прізвище, ім’я, по батькові)

Київ 2019

1. Ціль та задачі

Целью работы является изучение основных правил написания рекурсивных функций в функциональном языке и изучение основных методов разработки функциональных программ с позиций Строго Функционального Языка.

Основные задачи :

− На примере GNU Common Lisp"a (GCLisp'a) научиться формулировать условие завершения рекурсии, описывать формирование результата функции и новых значений аргументов для рекурсивного вызова;

− Получить практические навыки работы со списочными структурами в выбранной реализации языка Лисп;

− Освоить приемы нисходящего и восходящего проектирования функциональных программ;

− Научиться выделять основные и вспомогательные функции с учетом разбиения задачи на подзадачи;

− Овладеть приемами использования накапливающих параметров во вспомогательных функциях;

− Ознакомиться с упреждающим использованием результата вызова функции.

2. Завдання

Задание 1. Описать функцию в соответствии со своим вариантом задания из Таблицы 1, вариант выдает преподаватель.

Описать функцию, которая для заданного списка lst формирует список-результат путем объединения результата реверсирования lst, результата реверсирования хвоста lst, результата реверсирования хвоста хвоста lst и так далее. Пример : для списка ‘(1 2 3 4 5 6) результатом будет : ‘(6 5 4 3 2 1 6 5 4 3 2 6 5 4 3 6 5 4 6 5 6).

Задание 2. Написать программу сортировки списка методом Шелла. Вычисление последовательности шагов сортировки производится в соответствии с вариантом в Таблице 2.

Методом Р. Седжвика, рассмотренным в [6].

Задание 3. Написать программу сортировки [6] списка в соответствии с вариантом в таблице 3.

Сортировка простыми включениями.

Задание 4. Написать программу объединения двух отсортированных списков в один. При этом порядок сортировки в списке-результате должен сохраняться.

Задание 5. Написать программу в соответствии с заданием из Таблицы 4.

Написать программу, возвращающую T, если lst2 является подсписком lst1 глубины N. Элементами списка могут быть атомы и (или) списки любой глубины вложения.

3. Звіт

1. Проверяем списки на атомы, после чего сравниваем их поочерёдно:

```
(defun rev (l)
  (cond
    ((null l) '())
    (t (append (rev (cdr l)) (list (car l))))))

(defun task1 (lst)
  (cond
    ((null lst) lst)
    (t (append (rev lst) (task1 (cdr lst))))))

(print (task1 '(1 2 3 4)))

;; result: (4 3 2 1 4 3 2 4 3 4)
```

2. Shell sort

```lisp
(defun set-by-index (v i l)
  (cond
    ((eq i 0) (cons v (cdr l)))
    (t (cons (car l) (set-by-index v (- i 1) (cdr l))))))

(defun swap (i j l)
  (cond
    ((> i j) (swap j i l))
    (t (set-by-index (nth i l) j (set-by-index (nth j l) i l)))))

(defun step-even (i)
  (+ 1 (- (* 9 (expt 2 i)) (* 9 (expt 2 (/ i 2))))))

(defun step-odd (i)
  (+ 1 (- (* 8 (expt 2 i)) (* 6 (expt 2 (/ (+ i 1) 2))))))

(defun inc-step (i)
  (cond
    ((eq (mod i 2) 0) (step-even i))
    (t (step-odd i))))

(defun check-step (v s)
  (cond
    ((null v) t)
    ((> (* v 3) s) nil)
    (t t)))

(defun inc-steps (s &optional (i 0) steps)
  (cond
    ((check-step (car steps) s) (inc-steps s (+ i 1) (cons (inc-step i) steps)))
    (t (cdr steps))))

(defun sort-step-until (i step l &optional (j i))
  (cond
    ((< j 0) l)
    ((< (nth i l) (nth j l))
      (sort-step-until j step (swap i j l) (- j step)))
    (t (sort-step-until i step l (- j step)))))

```

```lisp
(defun sort-step (step l &optional (i 0))
  (cond
    ((≥ i (length l)) l)
    (t (sort-step step (sort-step-until i step l) (+ i step)))))

(defun sort-by-steps (steps l)
  (if
    (null steps)
    l
    (sort-by-steps (cdr steps) (sort-step (car steps) l))))


;;------------------------------------------------------------------------------
(defun shell-sort (l)
  (sort-by-steps (inc-steps (length l)) l))

(print (shell-sort '(10 9 8 7 6 5 4 3 2 1)))

```

3. Insertion sort:

```lisp
(defun insert (i l &optional (k #'<))
  (if (null l)
    (list i)
    (if (funcall k i (car l))
      (cons i l)
      (cons (car l) (insert i (cdr l) k)))))

(defun task3 (lst &optional (key #'<))
  (if (null lst)
    lst
    (insert (car lst) (task3 (cdr lst) key) key)
  )
)

(print (task3 '(3 2 4 4 1)))

;; result (1 2 3 4 4)
```

4.

```lisp
;; merge lists
(defun task4 (l1 l2 &optional res #'())
  (cond
    ((and (null l1) (null l2))
      res)
    ((null l1)
      (append res l2))
    ((null l2)
      (append res l1))
    ((< (car l1) (car l2))
      (task4 (cdr l1) l2 (append res (list (car l1)))))
    (t
      (task4 l1 (cdr l2) (append res (list (car l2)))))))

(print (task4 '(1 4 6 7) '(2 3 5 10)))

;; result (1 2 3 4 5 6 7 10)
```

5.

```lisp
;; sub list search
(defun task5 (l lst depth)
  (cond
    ((null (car lst)) nil)
    ((equal (car lst) l) t)
    ((eq 0 depth) nil)
    ((listp (car lst)) (task5 l (car lst) (- depth 1)))))

(print (task5 '(1 2 3) '(12) 4))
```

4. Висновки

В даній лабораторній роботі я вивчив спосіб написання рекурсивних функцій в мові GNU/Lisp та методику написання програм в строго функційній парадигмі. А саме:

- навчився правильно формувати вихід з рекурсії;
- навчився працювати з структурою даних "список";
- навчився виконувати задачу розбиттям останньої на підзадачі (з декларуванням допоміжних функцій);