Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Новгородский государственный университет имени Ярослава Мудрого»

Институт электронных и информационных систем

Кафедра информационных технологий и систем

**ОТЧЕТ**

по лабораторной работе № 3

“Методы разработки функциональных программ”

Разработал:

Студент группы 0092

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Авдеев Е. В.

«\_\_\_\_»\_\_\_\_\_\_\_\_2023 г.

Проверил преподаватель:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_Михайлов Д. В.

«\_\_\_\_»\_\_\_\_\_\_\_2023 г.

Великий Новгород

2023

1. Цели и задачи проводимого исследования
   1. Цель

Целью работы является изучение основных методов разработки функциональных программ с позиций строго функционального языка.

* 1. Задачи
* 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.

1. Задание 1

Написать программу сортировки списка методом Шелла (вычисляя шаг сортировки методом Р. Седжвика).

Код программы с комментариями:

Функция Седжвика, для вычисления шага сортировки:

(defun method\_sedjvik(s)

(if (= (mod s 2) 0)

(

+ (- (\* 9 (expt 2 s)) (\* 9 (expt 2 (floor s 2)))) 1

)

(

+ (- (\* 8 (expt 2 s)) (\* 6 (expt 2 (floor (+ s 1) 2)))) 1

)

)

)

Функция для вычисления коэффициентов для распределения:

(defun coefficient (s size last)

(cond

((>= last size) (if (> s 0) (- s 2) 0 ) )

(T

(coefficient (+ s 1) size (method\_sedjvik s))

)

)

)

Функция для вычисления списка шагов сортировки:

(defun steps\_list (s size last)

(cond

((>= last size) '())

(T

(append (list (method\_sedjvik s)) (steps\_list (+ s 1) size (method\_sedjvik s))

)

)

)

)

Функция принимает на вход список и индекс, возвращает элемент из списка, найденный по этому индексу:

(defun get\_list\_element (lst n)

(cond

((= n 0) (car lst))

(T

(get\_list\_element (cdr lst) (- n 1))

)

)

)

Функция заменяет элемент из списка по индексу:

(defun replace\_list\_element (lst n elem)

(cond

((= n 0) (cons elem (cdr lst)))

(T

(cons (car lst) (replace\_list\_element (cdr lst) (- n 1) elem))

)

)

)

Функция удаляет элемент из списка по индексу:

(defun remove\_list\_element (lst n)

(cond

((null lst) '())

((= n 0) (cdr lst))

(T

(cons (car lst) (remove\_list\_element (cdr lst) (- n 1)))

)

)

)

Функция, которая объединяет один элемент с другим:

(defun cons\_list(element lst)

(cond

((null lst) '())

(T

(cons

(cons element (car lst))

(cons\_list element (cdr lst))

)

)

)

)

Функция сортировки вставками:

(defun for\_last (lst j tmp inc)

(cond

((or (< j 0) (<= (get\_list\_element lst j) tmp)) (replace\_list\_element lst (+ j inc) tmp))

(T

(for\_last (replace\_list\_element lst (+ j inc) (get\_list\_element lst j)) (- j inc) tmp inc)

)

)

)

Функция для перебора с получением инкремента:

(defun for\_first (lst size i inc)

(cond

((>= i size) lst)

(T

(for\_first (for\_last lst (- i inc) (get\_list\_element lst i) inc) size (+ i 1) inc)

)

)

)

Функция для инициации алгоритма сортировки:

(defun start (lst size s slst)

(cond

((< s 0) lst)

(T

(start (for\_first lst size (get\_list\_element slst s) (get\_list\_element slst s)) size (- s 1) slst)

)

)

)

Инициализация сортировки:

(defun \_init (lst)

(cond

((null lst) 'Empty)

(T (start lst (length lst) (coefficient 0 (length lst) 0) (steps\_list 0 (length lst) 0)))

)

)

Результат выполнения:

Пример взят из (Приложение 1).  
Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

1. Задание 2

Написать программу сортировки списка простыми включениями.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Функция insert рекурсивно расставляет элементы списка. Если список пуст или элемент меньше первого элемента отсортированного списка, он просто сохраняет элемент в начале. В противном случае он сравнивает элемент с первым элементом и рекурсивно вставляет его в остальную часть списка.

Функция insertion-sort рекурсивно делит входной список на его первый элемент и остальную часть списка.

Результат работы

Изображение выглядит как текст, Шрифт, черный, типография

Автоматически созданное описание

1. Задание 3

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

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Основной алгоритм состоит в том, что проверить (< (car frt) (car sec)) - если первая запись первого списка меньше первой записи второго списка, то к результату добавляется первая запись первого списка и рекурсивно вызывается функция слияния с оставшейся частью первого списка и вторым списком.

Результат:



1. Задание 4

Написать программу, возвращающую Т, если lst2 является подсписком lst1

глубины N. Элементами списка могут быть атомы и (или) списки любой глубины

вложения

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Автоматически созданное описание

Функция **sublist-of-depth-N** использует комбинацию cond и рекурсивных вызовов для обхода списков и определения, является ли lst2 допустимым подсписком lst.

Результат  


Иначе

1. Заключение

Работа выполнена в полном объёме. Я освоил приёмы нисходящего и восходящего проектирования функциональных программ. Научился выделять основные и вспомогательные функции с учетом разбиения задачи на подзадачи. Овладел приемами использования накапливающих параметров во вспомогательных функциях. Ознакомился с упреждающим использованием результата вызова функции.

Приложение

Ссылка на Git репозиторий проекта - <https://github.com/gerundv/Lisp-Labs>

**К word файлу с отчётом к лабораторной работе №3 прилагаются**:

Файл task1.lisp – исходный код задания 1.

Файл task2.lisp – исходный код задания 2.

Файл task3.lisp – исходный код задания 3.

Файл task4.lisp – исходный код задания 4.

Литература:

1. Алгоритмы, методы, исходники: сортировка [Электронный ресурс]. URL:

http://algolist.manual.ru (дата обращения 05.06.2020).