**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**

Національний технічний університет України „КПІ”

Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки інформації та управління

**ЗВІТ**

з комп’ютерного практикуму № 1

з дисципліни «Мультипарадигменне програмування»

**Виконала**  студентка *3 курсу групи ІП-61*

*Кухарець Лілія Савеліївна*

*(номер залікової книжки 6115)*

Київ 2018

## ЗМІСТ

[ЦІЛЬ 3](#_Toc525671341)

[ПОСТАНОВКА ЗАДАЧІ 4](#_Toc525671342)

[ЛІСТИНГ ПРОГРАМИ 5](#_Toc525671343)

[РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ 10](#_Toc525671344)

[ВИСНОВКИ 13](#_Toc525671345)

## ЦІЛЬ

Метою роботи є вивчення основних правил написання рекурсивних функцій в

функціональній мові і вивчення основних методів розробки функціональних програм з позицій строго функціональної мови.

Основні завдання:

* На прикладі GNU Common Lisp'а (GCLisp'а) навчитися формулювати умови завершення рекурсії, описувати формування результату функції і нових значень аргументів для рекурсивного виклику;
* Отримати практичні навички роботи з обліковим структурами в обраній реалізації мови Лісп;
* Освоїти прийоми спадного і висхідного проектування функціональних програм;
* Навчитися виділяти основні і допоміжні функції з урахуванням розбиття завдання на підзадачі;
* Оволодіти прийомами використання накопичування параметрів у допоміжних функціях;
* Ознайомитися з випереджувальним використанням результату виклику функції.

## ПОСТАНОВКА ЗАДАЧІ

Завдання 1

Реалізувати функцію, яка міняє місцями перший і останній елементи

вихідного списку

Завдання 2

Написати програму сортування списку методом Шелла методом, запропонованим Дональдом Кнутом:

hk -1 = 3 \* h k + 1, ht= 1, де hi -крок сортування, t =[log3n]- 1- число кроків сортування, n - довжина списку.

Завдання 3

Написати програму сортування списку Шейкер-сортуванням.

Завдання 4

Написати програму об'єднання двох відсортованих списків в один. При цьому порядок сортування в списку-результаті повинен зберігатися.

Завдання 5

Задані глибина підсписка і позиція. Видалити з усіх наявних підсписков

заданої глибини елементи, що знаходяться на вказаній позиції.

## ЛІСТИНГ ПРОГРАМИ

1

(defun middle (list1 list2)

(if (atom(cdr list1)) list2

(middle (cdr list1) (append list2 (list(car list1)) ))

)

)

(defun change (list1)

(append (last list1) (middle (cdr list1) '( )) (list (car list1))))

(princ "1.Change 1st and last:")

(princ (change '(1 2 3 4 5 6 7 8 9)))

(terpri)

2

(defun findh (paramt h)

(if (or (eq paramt 1) (eq paramt 0)) h

(findh (- paramt 1) (+ 1 (\* 3 h)))

)

)

(defun len (list1 l)

(if (null list1) l

(len (cdr list1) (+ l 1))

)

)

(defun paste (a list1 list2)

(if (null list1) (append list2 (list a) list1)

(if (> (car list1) a) (append list2 (list a) list1)

(paste a (cdr list1) (append list2 (list (car list1) )))

)

)

)

(defun simplesort (list1 list2)

(if (null list1) list2

(simplesort (cdr list1) (paste (car list1) list2 '()) ))

)

)

(defun mymerge (list1 list2 list3 h position reqpos)

(if (null list2) (append list3 list1)

(if (not (eq (rem position h) reqpos)) (mymerge (cdr list1) list2 (append list3 (list(car list1))) h (+ position 1) reqpos)

(mymerge list1 (cdr list2) (append list3 (list(car list2))) h (+ position 1) reqpos)

)

)

)

(defun sortwithh (list1 list2 list3 h reqposition position)

(if (null list1) (mymerge list3 (simplesort list2 '()) '() h 0 reqposition)

(if (eq (rem position h) reqposition) (sortwithh (cdr list1) (append list2 (list (car list1))) list3 h reqposition (+ position 1))

(sortwithh (cdr list1) list2 (append list3 (list (car list1))) h reqposition (+ position 1))

)

)

)

(defun globalsortwithh (list1 h step reqposition)

(if (eq step 0) list1

(globalsortwithh (sortwithh list1 '() '() h reqposition 0) h (- step 1) (+ reqposition 1))

)

)

(defun shellsort2 (list1 h )

(if (eq h 1) (simplesort list1 '())

(shellsort2 (globalsortwithh list1 h h 0) (floor(/ (- h 1) 3)))

)

)

(defun shellsort (list1)

(shellsort2 list1 (findh (floor (- (log (len list1 0) 3) 1)) 1))

)

(print (shellsort '(1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3 1 2 3)))

3

(defun eonesort (list1 list2)

(if

(atom (cdr list1)) (append (last list1) list2 )

(if (> (car (last list1)) (car (last (butlast list1))))

(eonesort (butlast list1) (append (last list1) list2 ))

(eonesort (append (butlast (butlast list1)) (last list1)) (append (last (butlast list1)) list2 ))

)

)

)

(defun ebubblesort (list1 list2 list3)

(if

(atom (cdr list1)) (append list3 (list(car list1)) list2 )

(bbubblesort (cdr (eonesort list1 '())) list2 (append list3 (list (car(eonesort list1 '())))))

)

)

(defun bonesort (list1 list2)

(if

(atom (cdr list1)) (append list2 (last list1))

(if (< (car list1) (car (cdr list1)))

(bonesort (cdr list1) (append list2 (list (car list1))))

(bonesort (append (list(car list1)) (cddr list1)) (append list2 (list(car (cdr list1)))))

)

)

)

(defun bbubblesort (list1 list2 list3)

(if

(atom (cdr list1)) (append list3 (list(car list1)) list2)

(ebubblesort (butlast(bonesort list1 '())) (append (last(bonesort list1 '())) list2) list3)

)

)

(princ "3.Shakersort:")

(princ (bbubblesort '(3 5 3 8 9 1 3 4) '() '()))

(terpri)

4

(defun mergesort (list1 list2 list3)

(if(and (null list1) (null list2)) list3

(if(and (null list1) (not(null list2))) (mergesort list1 (cdr list2) (append list3 (list (car list2))))

(if(and (not(null list1)) (null list2)) (mergesort (cdr list1) list2 (append list3 (list (car list1))))

(if(< (car list1) (car list2)) (mergesort (cdr list1) list2 (append list3 (list (car list1))))

(mergesort list1 (cdr list2) (append list3 (list (car list2))))

)

)

)

)

)

(princ "4.One list from two sorted:")

(princ (mergesort '(1 3 6 9 10) '( 5 7 8) '()))

(terpri)

5

(defun len (list1 l)

(if (null list1) l

(len (cdr list1) (+ l 1))

)

)

(defun delelem (list1 list2 elem)

(if(eq (len list2 0) (- elem 1)) (append list2 (cdr list1) )

(delelem (cdr list1) (append list2 (list (car list1)) ) elem))

)

)

(defun deleting (list1 list2 depth elem)

(if

(null list1) list2

(if (atom (car list1)) (deleting (cdr list1) (append list2 (list (car list1))) depth elem)

(if (not (eq (len (car list1) 0) depth)) (deleting (cdr list1) (append list2 (list(deleting (car list1) '() depth elem))) depth elem)

(deleting (cdr list1) (append list2 (list(deleting (delelem(car list1) '() elem) '() depth elem))) depth elem)

)

)

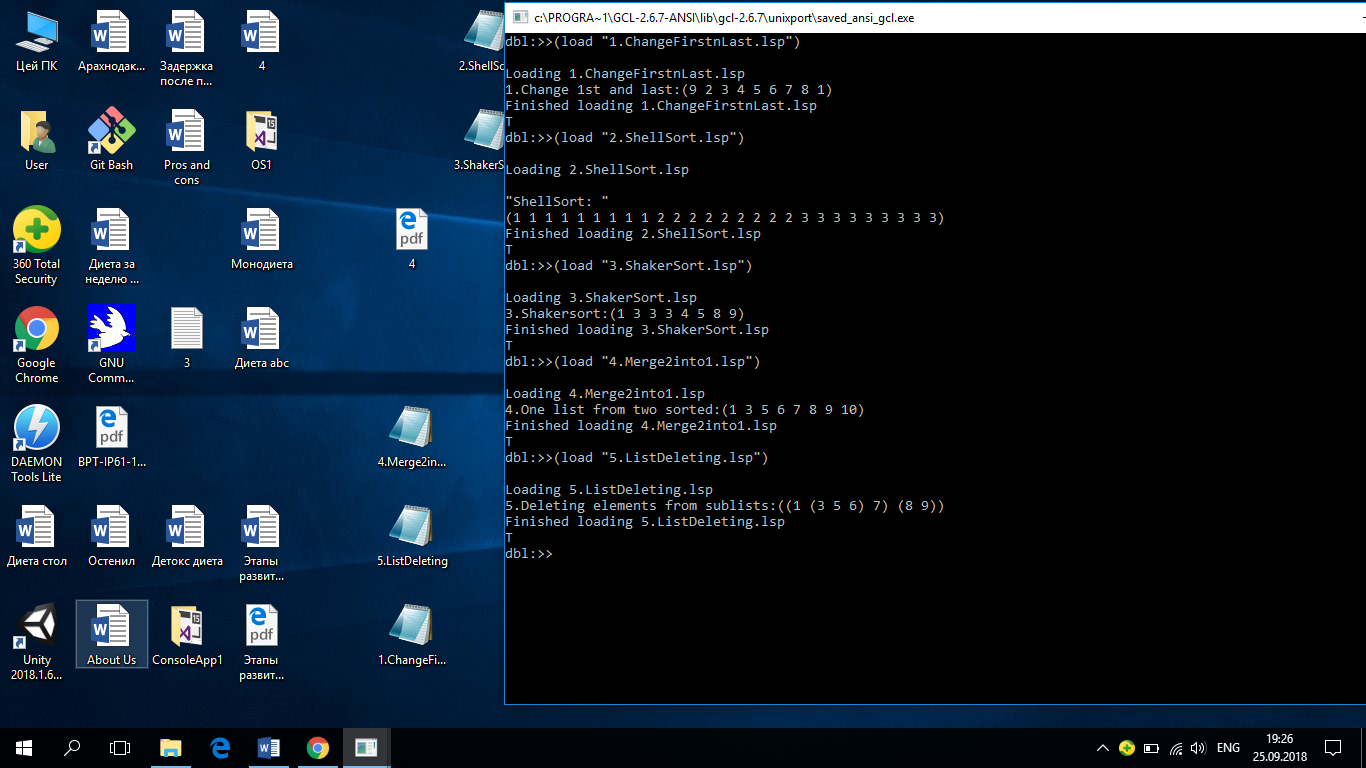
)

)

(princ "5.Deleting elements from sublists:")

(princ (deleting '((1 2 (3 4 5 6) 7) (8 9)) '() 4 2))

## РЕЗУЛЬТАТИ РОБОТИ ПРОГРАМИ



Функція change – об’єднує останній елемент списку, його середину та перший елемент списку.

Рекурсивна функція middle – повертає середину списку. Умова виходу – у першому списку лишився один елемент. Параметри: хвіст першого списку та об’єднання другого списку з першим елементом першого списку.

1. Shellsort – викликає рекурсивну функцію сортування.

shellsort2 – рекурсивна функція сортування. Умова виходу – крок сортування =1 (тоді виконується сортування з кроком 1). Параметри: результат виконання функції globalsortwithh, що приймає список1, поточний крок, кількість підмасивів =поточному кроку, номер підсписку, що сортується=0; змінений за допомогою формули поточний крок.

Globalsortwithh – функція сортування списку з заданим кроком. Умова виходу – кількість незмінених підмасивів=0. Зміна параметрів: відсортований з поточним кроком список (приймає параметри: даний список, два порожніх допоміжних списки, крок, номер підсписку, що сортується, поточна позиція=0), крок, номер не відсортованих підсписків, зменшений на 1, номер списка що сортується, збільшений на 1.

Sortwithh – функція сортування підсписку. Умова виходу – досягли кінця списку, тоді повертаємо з’єднані з допомогою mymerge відсортований з допомогою simplesort елементи поточного підсписку та інші елементи списку. Зміна параметрів: хвіст списку1, додаємо голову 1списку в кінець списку2, або списку 3, тоді 3 чи 2 список відповідно не змінюється, поточна позиція збільшується на 1.

Mymerge – функція об’єднання виділеного підсписку та елементів,, що лишились. Вихід – якщо список 2 закінчено (повертаємо об’єднання списку з результатом та першого (на випадок, якщо там ще лишились елементи)). Зміна параметрів – хвіст списку 1, список 2, об’єднання списку 3 і голови списку 1 (або список 1, хвіст 2, 3+голова2), поточна позиція зростає на 1.

Simplesort – функція звичайного сортування методом вставок з кроком 1. Умова виходу – кінець списку1. Зміна параметрів – хвіст списку1, список, що повертає ф-я paste, що приймає голову спису1 і список-рузультат2.

Paste – ставить елемент на потрібне місце у списку. Умова виходу – список1 закінчено або знайшли елемент, більший даного, тоді ставимо елемент в кінець списку-результату2 чи на потрібне місце. Зміна параметрів – хвіст списку1, список-результат + голова списку1.

Len – повертає довжину списку. Вихід – досягнуто кінця списку. Параметр – хвіст списку, збільшений на 1 розмір списку.

Findh – рекурсивно шукає початкове значення h. Умова виходу – t=1 або 0. Параметри – зменшене на 1 t, збільшене за допомогою формули h.

1. Bbubblesort, ebubblesort – функції бульбашкового сортування вперед і назад. Викликають одна одну рекурсивно. Умова виходу – початковий список скінчився.

Bonesort, eonesort – один прохід сортування вперед і назад. Умова виходу – у списку лишився 1 елемент, тоді додаємо його на початок або в кінець результуючого списку і повертаємо результат.

Зміна параметрів (для обміну сусідніх елементів за потреби): в першій функції хвіст початкового списку і результуючий+голова початкового або початковий без 2-го елементу і результуючий+передостанній елемент початкового. Для другої функції передаємо початковий без останнього або передостаннього елементу і відповідний елемент+ список-результат.

1. Mergesort- функція об’єднання 2 відсортованих списків в 1. Вихід – обидва списки закінчились. Параметри (для вставлення по черзі в список-результат найменшої голови початкових списків): або списки 1, хвіст 2, 3+голова 2, або хвіст 1, 2, 3+голова 1.
2. Deleting – основна функція для рекурсивного видалання елементів з підсписків. Умова виходу – початковий список закінчився. Зміна параметрів: якщо перший елемент списку на даному етапі – не список або його довжина не = заданій – хвіст списку, список-рузультат2 + голова початкового списку1; якщо довжина даного підсписку = заданій – хвіст початкового списку, список-результат + перший елемент списку1, до якого застосували delelem і Deleting.

Delelem – видалення вказаного елементу зі списку.

Умова виходу – номер поточного елемента мпівпав з номером для видалення. Зміна параметрів: хвіст списку1, список2+голова1.

## ВИСНОВКИ

У даній лабораторній роботі я навчилася працювати з рекурсією, обирати необхідні умови виходу та змінювати параметри відповідно до вимог задачі, спробувала розбивати задачу на декілька підзадач, у виклику однієї рекурсивної функції застосовувати виклики інших рекурсивних функцій, повторила деякі алгоритми сортування.