

Міністерство освіти та науки України
Національний технічний університет України “КПІ ім. Ігоря Сікорського”
Факультет інформатики та обчислювальної техніки
Кафедра автоматизованих систем обробки інформації і управління

ЗВІТ
про виконання
Лабораторної роботи №2
з дисципліни:
«Декларативне програмування»

Тема: «Опис простих рекурсивних функцій в мові Lisp»
Варіант 11

Виконав: студент групи ІП-52
Онбиш Олександр Олегович
Перевірів: доц. Баклан І. В.

Київ 2017

Цель работы:

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.

Задание 1.

Ознакомиться по лекционному материалу с описанием рекурсивных функций в Лиспе.

Выполнить примеры.

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

11.	Реализовать функцию, меняющую местами первый и последний элементы исходного списка.
-----	---

Код програми:

```
(defun swap (list)
  (if (null (rest list))
      list
      (cons (first (last list))
            (append (butlast (rest list))
                    (list (first list))))))
```

```
(print (swap '(1 2 3 4)))
(print (swap '(1 2)))
(print (swap '(1)))
```

Скріншот програми:

```
→ lab2 git:(master) X cat task_1.lisp
(defun swap (list)
  (if (null (rest list))
      list
      (cons (first (last list))
            (append (butlast (rest list))
                    (list (first list))))))

(print (swap '(1 2 3 4)))
(print (swap '(1 2)))
(print (swap '(1)))
→ lab2 git:(master) X clisp task_1.lisp

(4 2 3 1)
(2 1)
(1)
→ lab2 git:(master) X █
```

В цій задачі я створив функцію яка переставляє місцями перший та останій елемент списку.

3.2 Задание 3

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

6.	Сортировка Хоара.
----	-------------------

Код програми:

```
(defun list>= (a b)
  (cond
    ((or (null a) (null b)) nil)
    ((>= a (car b)) (list>= a (cdr b)))
    (t (cons (car b) (list>= a (cdr b))))))

(defun list< (a b)
  (cond
    ((or (null a) (null b)) nil)
    ((< a (car b)) (list< a (cdr b)))
    (t (cons (car b) (list< a (cdr b))))))

(defun hoare (L)
  (cond
    ((null L) nil)
    (t
     (append
      (hoare (list< (car L) (cdr L)))
      (cons (car L) nil)
      (hoare (list>= (car L) (cdr L))))))

(print (hoare '(1 5 3 8 2)))
(print (hoare '(9 8 7 6 5 4 3 2 1)))
(print (hoare '(1)))
```

Скріншот програми:

```
→ lab2 git:(master) X cat task_3.lisp
(defun list>= (a b)
  (cond
    ((or (null a) (null b)) nil)
    ((>= a (car b)) (list>= a (cdr b)))
    (t (cons (car b) (list>= a (cdr b))))))

(defun list< (a b)
  (cond
    ((or (null a) (null b)) nil)
    ((< a (car b)) (list< a (cdr b)))
    (t (cons (car b) (list< a (cdr b))))))

(defun hoare (L)
  (cond
    ((null L) nil)
    (t
     (append
      (hoare (list< (car L) (cdr L)))
      (cons (car L) nil)
      (hoare (list>= (car L) (cdr L)))))))

(print (hoare '(1 5 3 8 2)))
(print (hoare '(9 8 7 6 5 4 3 2 1)))
(print (hoare '(1)))
→ lab2 git:(master) X clisp task_3.lisp

(1 2 3 5 8)
(1 2 3 4 5 6 7 8 9)
(1)
→ lab2 git:(master) X █
```

В цьому завданні я дві функції, які розбивають список відносно опорного елементу і зарахунок них реалізує сортування Хоара, яка писується як рекурсивний вилик сортування Хоара для списків зліва та справа опорного елемента.

3.4 Задание 4

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

Код програми:

```
(defun merge-lists (a b)
  (cond ((not a) b)
        ((not b) a)
        ((< (car a) (car b)) (cons (car a) (merge-lists (cdr a) b)))))
```

```
(T (cons (car b) (merge-lists a (cdr b))))))
```

```
(print (merge-lists '(1 3 5 7) '(2 4 6 8 9)))
```

Скріншот програми:

```
→ lab2 git:(master) X cat task_4.lisp
(defun merge-lists (a b)
  (cond ((not a) b)
        ((not b) a)
        ((< (car a) (car b)) (cons (car a) (merge-lists (cdr a) b)))
        (T (cons (car b) (merge-lists a (cdr b))))))

(print (merge-lists '(1 3 5 7) '(2 4 6 8 9)))
→ lab2 git:(master) X clisp task_4.lisp
(1 2 3 4 5 6 7 8 9)
→ lab2 git:(master) X █
```

В цьому завданні реалізувати функцію злиття двох відсортованих списків за допомогою рекурсивного виклику злиття хвостів двох списків.

3.5 Задание 5

Написать программу в соответствии с заданием из Таблицы 4.

11, 23	Заданы глубина подписки и позиция. Удалить из всех имеющихся подписков заданной глубины элементы, находящиеся на указанной позиции.
--------	---

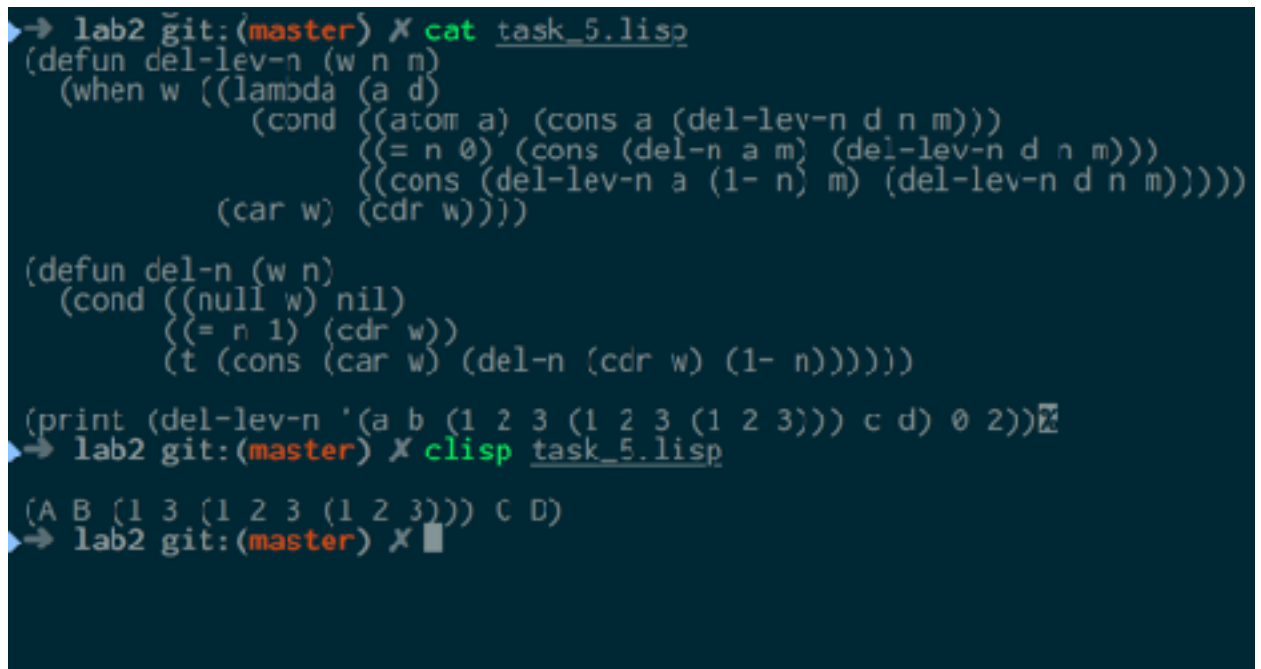
Код програми:

```
(defun del-lev-n (w n m)
  (when w ((lambda (a d)
              (cond ((atom a) (cons a (del-lev-n d n m)))
                    ((= n 0) (cons (del-n a m) (del-lev-n d n m)))
                    ((cons (del-lev-n a (1- n) m) (del-lev-n d n m))))))
    (car w) (cdr w))))
```

```
(defun del-n (w n)
  (cond ((null w) nil)
        ((= n 1) (cdr w))
        (t (cons (car w) (del-n (cdr w) (1- n))))))
```

```
(print (del-lev-n '(a b (1 2 3 (1 2 3 (1 2 3)))) c d) 0 2))
```

Скріншот програми:



```
→ lab2 git:(master) X cat task_5.lisp
(defun del-lev-n (w n m)
  (when w ((lambda (a d)
              (cond ((atom a) (cons a (del-lev-n d n m)))
                    ((= n 0) (cons (del-n a m) (del-lev-n d n m)))
                    ((cons (del-lev-n a (1- n) m) (del-lev-n d n m))))))
    (car w) (cdr w)))

(defun del-n (w n)
  (cond ((null w) nil)
        ((= n 1) (cdr w))
        (t (cons (car w) (del-n (cdr w) (1- n))))))

(print (del-lev-n '(a b (1 2 3 (1 2 3 (1 2 3)))) c d) 0 2))
→ lab2 git:(master) X clisp task_5.lisp

(A B (1 3 (1 2 3 (1 2 3))) C D)
→ lab2 git:(master) X
```

Висновок:

В цій лабораторній роботі я опанував базовий механізм виклику та опису рекурсивних функцій у інтерпретованій мові Lisp, навчився користуватися інтерпретатором GCL та завантажувати з нього файли з розширенням *.lisp, також опанував деякі продвинуті функції роботи зі списками.