**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 3

**З дисципліни:** *“Декларативне програмування”*

**На тему:** *“Підвищення ефективності функційних програм”*

**Лектор:**

доц. каф. ПЗ

Левус Є. В.

**Виконав:**

ст. гр. ПЗ-44

Вонс Ю. А.

**Прийняв:**

асист. каф. ПЗ

Цимбалюк Т.М.

« \_\_\_\_ » \_\_\_\_\_\_\_\_ 2019 р.

∑= \_\_\_\_\_ .

Львів – 2019

**Тема роботи:** Підвищення ефективності функційних програм.

**Мета роботи:** Навчитися створювати функції на основі базових примітивів, використовуючи метод параметрів нагромадження.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

Поняття ефективності програм пов’язано з якістю програмного забезпечення і визначається кількістю обчислювальних ресурсів, необхідних для виконання програми. Прийнято вважати, що задача підвищення ефективності програм розв’язана, якщо вдалося зменшити кількість обчислювальних ресурсів на 25 відсотків. Функційна програма складається з множин підфункцій, які визначені одна через другу. Хвостова рекурсія є спеціальним видом рекурсії, в якій є єдиний виклик рекурсивної функції і при цьому цей виклик виконується після всіх обчислень.

При реалізації обчислення хвостової рекурсії можуть виконуватися за допомогою ітерацій в постійному обсязі пам'яті. Пам'ять витрачається лише на збереження адрес повернення значень функції. На практиці це означає, що «хороший» транслятор функційної мови повинен «вміти» розпізнавати хвостову рекурсію і реалізовувати її у вигляді циклу. У свою чергу, метод параметра нагромадження не завжди призводить до хвостової рекурсії, проте він однозначно допомагає зменшити загальний обсяг пам'яті.

Принципи побудови функцій з параметром нагромадження:

1. Вводиться нова функція з додатковим параметром (акумулятором), в котрому накопичуються результати обчислень;
2. Початкове значення акумулятора аргументу задається у рівності, що пов'язує стару і нову функції;
3. Ті рівності вихідної функції, які відповідають виходу з рекурсії, замінюються поверненням акумулятора;
4. Рівності, що відповідають рекурсивному виклику, виглядають як звернення до нової функції, в якому акумулятор отримує те значення, яке повертається вихідною функцією.

**ЗАВДАННЯ**

Варіант №3:

Визначити функцію РІВЕНЬ(Х), яка обертає список і розбиває його на рівні: (a, b, c) = (((c) b) a).

**ХІД ВИКОНАННЯ**

(define (func\_size arg)

(cond

((eq? arg '()) 0)

(#t (+ (func\_size (cdr arg)) 1))))

(define (func\_list arg1 arg2)

(cons arg1 (cons arg2 '())))

(define (func\_level arg)

(cond

((eq? arg '()) '())

((eq? (func\_size arg) 1) cdr arg)

(#t (func\_list (func\_level (cdr arg))(car arg)))))

(define (func\_empty? arg)

(cond

((> (func\_size arg) 0) 0)

(#t 1)))

(define (func\_pair? arg)

(cond

((pair? arg) 0)

(#t 1)))

(define (func\_verify\_list arg)

(cond ((eq? arg '()) '())

((pair? (cdr arg)) (cons (car arg) (func\_verify\_list (cdr arg))))

((list? (cdr arg)) arg)

(#t (cons arg '()))))

(define (atom? arg)

(cond

((list? arg) '#f)

((pair? arg) '#f)

(#t '#t)))

(define (type arg)

(cond

((list? arg)

(cond

((> (func\_size arg) 0) 'NOT\_EMPTY\_LIST)

((eq? (func\_size arg) 0) 'EMPTY\_LIST)))

((pair? arg) (cond ((atom? (cdr arg)) 'PAIR)

((pair? (cdr arg)) 'NOT\_CORRECT\_LIST)

(#t 'PAIR)))

(#t 'ATOM)))

(define (level arg)

(cond

((eq? (atom? arg) '#t) "Атом")

((= (func\_empty? (func\_verify\_list arg)) 1) "Пустий список")

((eq? (TYPE arg) 'PAIR) "Пара")

((< (func\_size (func\_verify\_list arg)) 2) "У спискку менше двох елементів")

(#t (func\_level (func\_verify\_list arg)))))

**РЕЗУЛЬТАТИ**

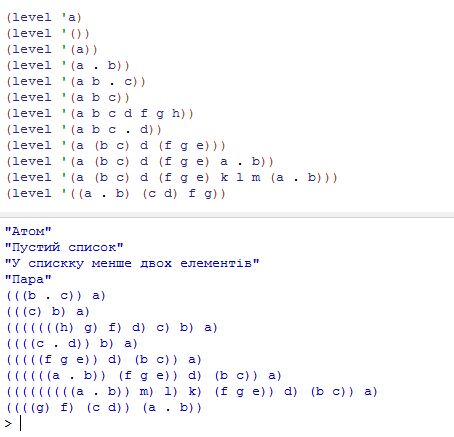


Рис. 1. Результат виконання програми

**ВИСНОВКИ**

Виконуючи дану лабораторну роботу, я ознайомився та навчився створювати функції на основі базових примітивів, використовуючи метод параметрів нагромадження. Також, я ознайомився з поняттями ефективності програми, “дорогих” функцій, параметрів-акумуляторів.

Під час виконання лабораторної роботи, я створив функцію РІВЕНЬ, яка обертає список і розбиває його на певні рівні, об’єднуючи списку у новий список, використовуючи метод параметрів нагромадження. Для реалізації даної функції, я використав додаткову функцію func\_level, яка об’єднує новий параметр – список із уже існуючим у цій функції. Також, у даній програмі було розроблено функції для перевірки на атом, пару, список та функцію для знаходження розміру списку.