**Міністерство Освіти І НАУКИ України**

**Національний університет "Львівська політехніка"**

Інститут **КНІТ**

Кафедра **ПЗ**

### ЗВІТ

До лабораторної роботи № 2

**З дисципліни:** *“Декларативне програмуваннях”*

**На тему:** *“Використання рекурсії у функціональному програмуванні”*

**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:** Використання рекурсії у функціональному програмуванні.

**Мета роботи:** Навчитися створювати функції на основі базових примітивів, використовуючи принцип рекурсії.

**TЕОРЕТИЧНІ ВІДОМОСТІ**

Рекурсивний алгоритм − це алгоритм, в описі якого прямо або непрямо міститься звернення до самого себе. Рекурсивний алгоритм завжди розбиває задачу на частини та класифікується, залежно від того, які функції можна визначити і обчислити з використанням різних форм рекурсії.

Рекурсивну тріаду складають параметризація, виділення бази і декомпозиція. На етапі параметризації з постановки задачі виділяються параметри, які описують вихідні дані. Виділення бази рекурсії передбачає перебування в розв'язуваної задачі тривіальних випадків, результат для яких очевидний і не потребує проведення розрахунків. Декомпозиція – це процес послідовного розбиття задачі на підзадачі.

Рекурсивні алгоритми відносяться до класу алгоритмів з високою ресурсомісткістю, так як при великій кількості самовикликів рекурсивних функцій відбувається швидке заповнення стекової області. Важливою характеристикою рекурсивного алгоритму є глибина рекурсивних викликів – найбільше одночасне кількість рекурсивних звернень функції.

Переваги рекурсії:

* простота і компактність запису алгоритмів;
* рекурсивна процедура показує, що потрібно робити, а нерекурсивна – як потрібно робити;
* легко програмувати обчислення за рекурентними формулами;
* легко доводиться коректність програми – її еквівалентність тим формулам, за якими розв’язується задача.

Недоліки рекурсивних визначень:

* неефективне витрачання оперативної пам’яті;
* низька швидкодія роботи програм.

**ЗАВДАННЯ**

**Варіант № 3.** Визначити функцію ОСТАННІЙ(Х), яка видає як результат останній елемент списку Х.

**ХІД ВИКОНАННЯ**

(DEFINE (FUNC\_EMPTY? arg)

(COND

((> (Func\_Size arg) 0) 0)

(#t 1)))

(DEFINE (Func\_Size arg)

(COND

((EQ? arg '()) 0)

(#t (+ (Func\_Size (CDR arg)) 1))))

(DEFINE (Func\_Contain\_Atom? arg)

(COND

((LIST? arg) 0)

((PAIR? arg) 0)

(#t 1)))

(DEFINE (Func\_Contain\_Pair arg)

(COND

((= (Func\_Size arg) 1)

(PAIR? arg))

(#t #f)))

(DEFINE (Func\_Search\_Last arg)

(COND

((EQ? arg '()) '())

((EQ? (CDR arg) '()) (CAR arg))

(#t (Func\_Search\_Last (CDR arg)))))

(DEFINE (Func\_Combine arg)

(COND

((EQ? arg '()) '())

((LIST? (CDR arg)) arg)

((PAIR? (CDR arg))

(CONS (CAR arg)(Func\_Combine (CDR arg))))

(#t (CONS arg '()))))

(DEFINE (LAST X)

(COND

((> (Func\_Contain\_Atom? X) 0) "Один атом")

((= (FUNC\_EMPTY? (Func\_Combine X)) 1) "Пустий список")

((Func\_Contain\_Pair (Func\_Combine X)) "Одна пара")

(#t (Func\_Search\_Last (Func\_Combine X)))))

**РЕЗУЛЬТАТИ**

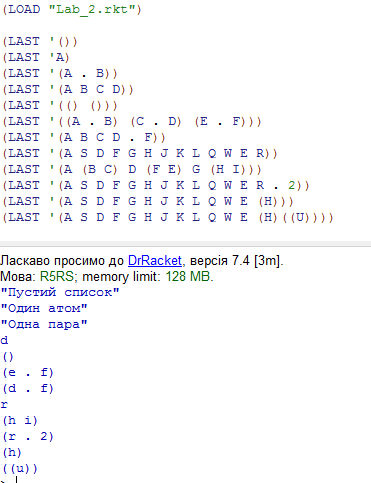


Рис. 1. Результат виконання програми

**ВИСНОВКИ**

Виконуючи дану лабораторну роботу, я ознайомився із основами застосування рекурсії в ітеративних процесах у функціональних мовах програмування та навчився створювати функції на основі базових примітивів, використовуючи принцип рекурсії. Результатом лабораторної роботи є те, що я створив програму із головною функцією Last, що видає як результат останній елемент списку. Для коректності виконання були створені термінальні гілки, у порядку від простої умови завершення до більш складної. У програмі розроблена рекурсивні функції для з’єднання елементів списку та для пошуку останнього елементу. Рекурсія у функції Combine є простою, прямою та за значенням. Також, рекурсія у функції Search\_Last є простою, прямою та за значенням. Ці рекурсії є простими оскільки одиничний виклик функції самої себе зустрічається в одній гілці, рекурсії є прямими оскільки всередині тіла цих функцій міститься виклик тієї ж функції та рекурсії є за значенням, оскільки виклик функції формує результат функції. Також, у програмі розроблено функції розміру списку, чи список є пустий, чи є атом у списку, чи є пара у списку.