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 № 4

“Локальные определения”

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 практическое изучение различных видов локальных определений и особенностей их использования в рекурсивных программах.

* 1. Задачи
* изучить применение техники нисходящей и восходящей рекурсии при написании рекурсивных функций с использованием локальных определений;
* сравнить возможности локальных определений LET и LAMBDA по организации вычислений в рекурсивных программах;

1. Задание 1

Описать функцию вычисления факториала. Рассмотреть варианты решения задачи с применением локальных определений LAMBDA и LET.

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

Автоматически созданное описание

Результат выполнения:

Факториал 5

![Изображение выглядит как текст, черный, Шрифт, белый

Автоматически созданное описание]()

1. Задание 2

Разработать программу символьного дифференцирования в соответствии с правилами, изложенными в [8]. Рассмотреть варианты решения задачи с применением локальных определений LAMBDA и LET.

Алгоритм был адаптирован из примера реализации в интернете:

(defun \_exponent (base exponent)

(cond

((= exponent 0) 1)

((= exponent 1) base)

(T (list '\*\* base exponent))

)

)

(defun is\_exponent (x)

(and (listp x) (eq (car x) '\*\*))

)

(defun is\_number\_and\_equal (exp num)

(and (numberp exp) (= exp num))

)

(defun is\_variable (x)

(symbolp x)

)

(defun variable\_is (v1 v2)

(and

(is\_variable v1)

(is\_variable v2)

(eq v1 v2)

)

)

(defun make\_product (m1 m2)

(cond

((or (is\_number\_and\_equal m1 0) (is\_number\_and\_equal m2 0)) 0)

((is\_number\_and\_equal m1 1) m2)

((is\_number\_and\_equal m2 1) m1)

((and (numberp m1) (numberp m2)) (\* m1 m2))

(T (list '\* m1 m2))

)

)

(defun is\_product (x)

(and (listp x) (eq (car x) '\*))

)

(defun make\_sum (a1 a2)

(cond

((is\_number\_and\_equal a1 0) a2)

((is\_number\_and\_equal a2 0) a1)

((and (numberp a1) (numberp a2)) (+ a1 a2))

(T (list '+ a1 a2))

)

)

(defun is\_sum (x)

(and (listp x) (eq (car x) '+))

)

(defun deriv (exp var)

(cond

((numberp exp) 0)

((is\_variable exp) (if (variable\_is exp var) 1 0))

((is\_sum exp) (make\_sum (deriv (cadr exp) var) (deriv (caddr exp) var)))

((is\_product exp)

(make\_sum

(make\_product (cadr exp) (deriv (caddr exp) var))

(make\_product (deriv (cadr exp) var) (caddr exp))

)

)

((is\_exponent exp)

(make\_product

(make\_product (caddr exp)

(\_exponent (cadr exp) (- (caddr exp) 1))

)

(deriv (cadr exp) var)

)

)

(T (error "Error type" exp))

)

)

(terpri)

(write (deriv '(\*\* x 3) 'x))

(terpri)

Результат выполнения:

Пример: '(\*\* x 3) 'x

Изображение выглядит как Шрифт, рукописный текст, черный, каллиграфия

Автоматически созданное описание

1. Задача 3

Решить задачу из лабораторной работы №2 с применением локальных определений LAMBDA и LET.

Задача из 2-й лабораторной работы. Описать функцию, которая для заданного списка lst формирует список-результат путем объединения результата реверсирования lst, результата реверсирования хвоста lst, результата реверсирования хвоста хвоста lst и так далее. Пример : для списка ‘(1 2 3 4 5 6) результатом будет : ‘(6 5 4 3 2 1 6 5 4 3 2 6 5 4 3 6 5 4 6 5 6).

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

Автоматически созданное описание

Функция reverse-tail рекурсивно обрабатывает хвост списка, реверсируя его и объединяя с предыдущими реверсированными хвостами. append используется для объединения списков.

Результат выполнения:

Изображение выглядит как текст, Шрифт, черный, Графика

Автоматически созданное описание

1. Задача 4

Реализовать простейший интерпретатор лисповских программ. На вход интерпретатора подается текст, который может быть интерпретирован как вызов или суперпозиция функций Лиспа, пример (для Common Lisp): ‘(cons (car (cdr ‘(e r t w))) (cons (cdr ‘(g h 6)) ‘())). Программа должна обеспечивать выполнение такого рода примеров. Требования к программе: − интерпретация базовых функций Лиспа и арифметических операций +, -, /, \*; − в программе должны использоваться локальные определения; − не допускается использование встроенной функции-интерпретатора EVAL;

Пример интерпретатора:

(defun interpret (expr)

(cond

((not (symbolp (car expr))) expr)

((not (fboundp (car expr))) expr)

(T

(

(lambda (m n)

(apply m n)

)

(car expr)

(arg\_proc (cdr expr))

)

)

)

)

(defun arg\_proc (lst)

(cond

((null lst) nil)

((atom (car lst)) (cons (car lst) (arg\_proc (cdr lst))))

((equal (symbol-name (caar lst)) "QUOTE") (cdar lst))

(T

(cons

(interpret (car lst))

(arg\_proc (cdr lst))

)

)

)

)

(write (interpret '(cons (car (cdr '(e r t w))) (cons (cdr '(g h 6)) '()))))

Результат выполнения:



1. Задание 5

Дополнить интерпретатор из 4-го задания функцией возведения в степень.

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

Автоматически созданное описание

Результат выполнения:



1. Заключение

Работа выполнена в полном объёме. Я изучил на практике различные виды локальных определений и особенностей их использования в рекурсивных программах. Изучил применение техники нисходящей и восходящей рекурсии при написании рекурсивных функций с использованием локальных определений. Сравнил возможности локальных определений LET и LAMBDA по организации вычислений в рекурсивных программах.

Приложение

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

**К word файлу с отчётом к лабораторной работе №4 прилагаются**:

Файл task1.lisp – код задания 1.

Файл task2.lisp – код задания 2.

Файл task3.lisp – код задания 3.

Файл task4.lisp – код задания 4.

Файл task5.lisp – код задания 5.