**Лабораторная работа №2.**

**Тема:** Определение функций. Функции ввода-вывода. Вычисления, изменяющие структуру.

**Цель:** Получить навыки в написании функций. Изучить функции ввода-вывода.

*1. Функции, определяемые пользователем.*

Определение функций и их вычисление в Лиспе основано на лямбда-исчислении Черча.

В Лиспе лямбда-выражение имеет вид

(LAMBDA (x1 x2 ... xn) fn)

Символ LAMBDA означает, что мы имеем дело с определением функции. Символы xi являются формальными параметрами определения, которые имеют аргументы в описывающем вычисления теле функции fn. Входящий в состав формы список, образованный из параметров, называют лямбда-списком.

Телом функции является произвольная форма, значение которой может вычислить интерпретатор Лиспа.

\_(lambda (x y) (+ x y))

Формальность параметров означает, что их можно заменить на любые другие символы, и это не отразится на вычислениях, определяемых функцией.

Лямбда-выражение - это определение вычислений и параметров функции в чистом виде без фактических параметров, или аргументов. Для того, чтобы применить такую функцию к некоторым аргументам, нужно в вызове функции поставить лямбда-определение на место имени функции:

(лямбда-выражение а1 а2 ... аn)

Здесь ai - формы, задающие фактические параметры, которые вычисляются как обычно.

\_((lambda (x y) (+ x y)) 1 2) ⇨ 3

Лямбда-вызовы можно свободно объединять между собой и другими формами. Вложенные лямбда-вызовы можно ставить как на место тела лямбда-выражения, так и на место фактических параметров.

\_((lambda (x) ;Тело лямбда-вызова -

((lambda (y) (list x y)) ‘b)) ‘a) ⇨ (a b) лямбда-вызов.

Записывать вызовы функций полностью с помощью лямбда-вызовов не разумно, поскольку очень скоро выражения в вызове пришлось бы повторять, хотя разные вызовы одной функции отличаются лишь в части фактических параметров. Проблема разрешима путем именования лямбда-выражений и использования в вызове лишь имени.

Дать имя и определить новую функцию можно с помощью функции DEFUN:

(DEFUN имя лямбда-список тело)

DEFUN соединяет символ с лямбда-выражением, и символ начинает представлять определенные этим лямбда-выражением вычисления. Значением этой формы является имя новой функции.

После именования функции ее вызов осуществляется по имени и параметрам.

\_(defun list1 (x y)

(cons x (cons y nil))) ⇨ list1

\_(list1 ‘c ‘n) ⇨ (c n)

(eval <выражение>)

Функция возвращает результат выражения <выражение>, где <выражение> - любое выражение языка LISP. Например, дано:

(setq a 123)

(setq b 'a)

(eval 4.0) ⇨ 4.000000

(eval (abs -10)) ⇨ 10

(eval a) ⇨ 123

(eval b) ⇨ 123

*2. Функция ввода.*

Лисповская функция чтения READ обрабатывает выражение целиком. Вызов функции осуществляется в виде

### \_(READ)

(Вводимое выражение) ⇨ ;выражение пользователя

⇨ (ВВОДИМОЕ ВЫРАЖЕНИЕ) ;значение функции READ

...

Функция не показывает, что она ждет ввода выражения. Она лишь читает выражение и возвращает в качестве значения само это выражение, после чего вычисления продолжаются.

Если прочитанное выражение необходимо сохранить для дальнейшего использования, то вызов READ должен быть аргументом какой-нибудь формы, например присваивания (SETQ), которая свяжет полученное выражение:

\_(SETQ input (READ))

(+ 1 2) ;введенное выражение

(+ 1 2) ;значение

\_input ⇨ (+1 2)

*3. Функции вывода.*

Для вывода выражений используют несколько функций: PRINT, PRIN1, PRINC.

Функция PRINT.

Это функция с одним аргументом, которая сначала вычисляет значение аргумента, а затем выводит это значение. Функция PRINT перед выводом аргумента переходит на новую строку, а после него выводит пробел. Таким образом, значение выводится всегда на новую строку.

\_(PRINT (+ 1 2))

3 ;вывод

3 ;значение

PRINT является псевдофункцией, у которой есть как побочный эффект, так и значение. Значением функции является значение ее аргумента, а побочным эффектом - печать этого значения.

Функции PRIN1 и PRINC.

Эти функции работают так же, как PRINT, но не переходят на новую строку и не выводят пробел:

(PRIN1 5) ⇨ 55

(PRINC 4) ⇨ 44

Обеими функциями можно выводить кроме атомов и списков и другие типы данных, которые мы рассмотрим позже:

\_(PRIN1 «CHG») ⇨ «CHG»«CHG»

\_(PRINC «tfd») ⇨ tfd«tfd» ;вывод без кавычек,

;результат - значение аргумента

С помощью функция PRINC можно получить более приятный вид. Она выводит лисповские объекты в том же виде, как и PRIN1, но преобразует некоторые типы данных в более простую форму.

Функция TERPRI.

Эта функция переводит строку. У функции TERPRI нет аргументов и в качестве значения она возвращает NIL:

\_(DEFUN out (x y)

(PRIN1 x) (PRINC y)

(TERPRI) (PRINC (LIST ‘x ‘y)) ⇨ out

\_(out 1 2) ⇨ 12

(1 2)

4. Вычисления, изменяющие структуру.

Основными функциями, изменяющими физическую структуру списков, являются RPLACA и RPLACD, которые уничтожают прежние и записывают новые значения в поля CAR и CDR списочной ячейки:

(RPLACA ячейка значение-поля) ;поле CAR

(RPLACD ячейка значение-поля) ;поле CDR

Первым аргументом является указатель на списочную ячейку, вторым - записываемое в нее новое значение поля CAR или CDR. Обе функции возвращают в качестве результата указатель на измененную списочную ячейку:

\_(SETQ a ‘(b c d)) ⇨ (b c d)

\_(RPLACA a ‘d) ⇨ (d c d)

\_(RPLACD a ‘(o n m)) ⇨ (d o n m)

\_a ⇨ (d o n m)

5. Задания к лабораторной работе.

5.1. Написать неименованные функции для задания 1 первой лабораторной работы

5.2. Написать две именованные функции, одну по заданию 2 первой лабораторной работы, задание для второй в таблице 1

Таблица 1

|  |  |
| --- | --- |
| Вариант | Задание |
| 1 | Есть два списка. Если первый элемент первого списка является атомом, то возвратить второй список, иначе возвратить список из головы первого и хвоста второго списков. |
| 2 | Написать функцию, которая возвращает квадрат аргумента, если аргумент является числом; хвост списка, если аргумент является списком и сам аргумент иначе. |
| 3 | Есть четыре числа. Написать функцию, формирующую список, содержащий максимальное и минимальное из этих чисел. |
| 4 | Написать функцию, формирующую список (X^3 + 1/X +2\*X) если X < 0, список (X^2 + 1/2\*X+ X^1/2), если X > 0, nil иначе. |
| 5 | Написать функцию, которая для заданных координат X1,Y1 и X2,Y2 возвращает расстояние между ними. Координаты могут иметь и отрицательные значения |
| 6 | Написать функцию, которая по заданному вещественному числу формирует список двух элементов. Первый элемент списка - это знак числа, второй принадлежность числа диапазону -5...+5. (второй аргумент может принимать значения T или NIL) |
| 7 | Написать функцию, которая по двум числам формирует список из двух элементов. Первый элемент - это результат целочисленного деления первого числа на второе, второй - остаток. Деление выполнять только, если делитель больше 0. |
| 8 | Есть список lst. Сформировать список из первого, второго и последнего элементов, если первый элемент число и из пятого, шестого и восьмого элементов иначе. |
| 9 | Написать функцию трех аргументов X Y Z, возвращающую X\*Y^3-Y^2 если Z > 0 и X^2 + Z^2 иначе. |
| 10 | Есть два списка, написать функцию, формирующую новый список по правилу: если второй элемент второго списка атом, то построить результирующий список из головы первого и хвоста второго списков, и из головы первого и головы второго списков иначе. |
| 11 | Есть два трехэлементных списка, к примеру '(1 2 3) и ( a b c), сформировать список вида '(1 a 2 b 3 c), в котором элементы исходных списков чередуются. |
| 12 | Есть два списка, написать функцию, формирующую список из двух подсписков. Первый подсписок состоит из первого элемента первого списка и последнего элемента второго списка, второй подсписок состоит из третьих элементов списков. |
| 13 | Есть списки, к примеру, '(1 2 3 4 5) '(7 6 5 7 ). Если произведение первых элементов положительное число, то объединить в результирующий список последние элементы списков - (5 7), если произведение первых элементов - отрицательное число, то объединить хвосты списков – (2 3 4 5 6 5 7) |
| 14 | Есть список, сформировать новый список из первых трех элементов, если второй элемент аргумента - атом, и из первых пяти иначе. Выполнить проверку, что исходный список содержит не менее пяти элементов, если меньше – результат NIL. |
| 15 | Есть список и объект (s-выражение). Написать функцию, возвращающую новый список, в котором объект замещает первый элемент списка, если первый элемент атом и второй элемент списка иначе. |
| 16 | Есть список, состоящий из чисел. Написать функцию возвращающую список из пяти первых удвоенных элементов, если первый элемент четный. Если первый элемент нечетный, то возвращаемый список должен содержать пять первых членов списка, каждый из которых уменьшен на половину. Выполнить проверку, что исходный список содержит не менее пяти элементов, если меньше – результат NIL. |
| 17 | Есть список и два объекта (s-выражения). Написать функцию, возвращающую исходный список, если оба объекта числа и их сумма больше 100, иначе функция должна возвратить список из объектов. |

5.3. Определите функции (NULL x), (CADDR x) и (LIST x1 x2 x3) с помощью базовых функций. Используйте имена NULL1, CADDR1 и LIST1, чтобы не переопределять одноименные встроенные функции системы.

5.4. Напишите функции, которые осуществляют обращение списка из 2, 3, 4 элементов.

5.5. Постройте функции, которые проверяют, является ли ее аргумент:

a) списком из 2, 3, 4 элементов;

b) списком из 2, 3, 4 атомов.

5.6. Напишите функцию, которая вводит фразу на естественном языке и преобразует ее в список. Например: «я пошел гулять» --- (я пошел гулять)

5.7. Напишите функцию, которая спрашивает у пользователя ФИО студента из группы (список группы составлен в лабораторной работе №1) и выдает следующие данные о нем:

1. год рождения;
2. средний бал;
3. родителей;
4. списки свойств, присвоенные ему раньше.

5.8. Напишите функцию:

1. от одного аргумента (ФИО), замещающую в списке с данными о нем подсписки со средними балами на списки свойств;
2. вычисляющую средние балы, беря данные из списков свойств.

6. Вопросы.

1. Что такое лямбда-выражение?

2. Для чего используется функция DEFUN?

3. Чем различаются основные функции вывода?

4. Что возвращает в качестве значения функция READ?

5. Особенности функций, изменяющих структуру?