ЛАБОРАТОРНАЯ РАБОТА 5

Описание и вызов функций в языке Лисп.

1. Цель и задачи. Целью работы является изучение базовых функций организации и обработки списков, а также способов описания и вызова нерекурсивных функций в языке программирования Лисп (на примере одного из известных диалектов языка Лисп).

Основные задачи :

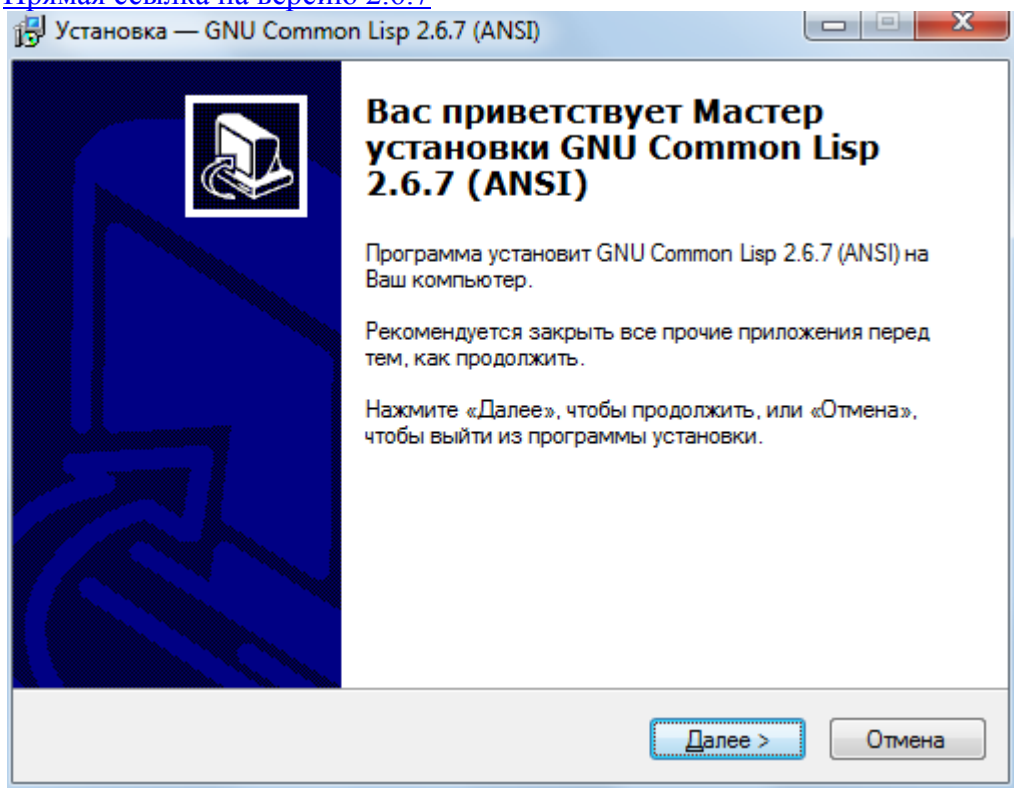
− Получить навыки работы с интерпретатором Лиспа для выбранного диалекта.
− Изучить работу примитивных базовых функций списочного ассемблера.
− Изучить работу базовых функций из расширения набора примитивных функций и их сведение к примитивным базовым функциям.
− Ознакомление с описанием неименованных функций в Лиспе.
− Изучение приемов описания именованных функций через неименованные и с применением современной сокращенной нотации.

2. Работа с интерпретатором Лиспа. Предусмотрена возможность выполнения лабораторного практикума в среде GCL (GNU Common Lisp).

2.1 Работа в среде Microsoft GNU Common Lisp.

Версию для Windows можно скачать по этому адресу

Прямая ссылка на версию 2.6.7



Примеры :
$(* 4 5)- вызов функции умножения для двух аргументов;
$(* 5 6 7 8)- тоже для четырех аргументов;
$(- 12 3)- вызов функции вычитания 12-3

Для выхода из среды Microsoft GNU Common Lisp необходимо набрать в командной строке интерпретатора :
$(QUIT).

2.1.1 Рекомендуемая последовательность действий при работе с интерпретатором.

− разработать текст программы;
− записать разработанный текст в файл с расширением .lsp.
− вызвать интерпретатор с указание выполняемого файла.

Для сокращения объема программы в первой лабораторной работе разрешается использовать псевдофункцию, связывающую имена и значения (SETQ имя значение). К примеру, нужно выполнять многократную обработку списка '(a c d (e f)). Обозначим этот список l1 и сделаем это так (SETQ l1 '(a c d (e f))). После такого связывания там, где используется значение списка, можно использовать имя l1.

2.1.2 Рекомендуемая структура программы.

− связывание имен и значений;
− вызовы функций, либо описание функций, т.е. собственно текст программы;
− функция переключения входного потока (RDS)

3. Базовые функции Лиспа.

Примитивные (простейшие) базовые функции обработки можно сравнить с основными арифметическими действиями: они просты и их мало.

Назначение этих функций состоит в:
− расчленении S-выражений (селекторы);
− составлении S-выражений (конструкторы);
− анализ S-выражений (предикаты).

3.1 Реализация базовых функций в GNU Common Lisp'e.

Таблица 1. Примитивные базовые функции.

Назначение.	Вызов.	Результат.
Селектор	(CAR список)	S-выражение
Селектор	(CDR список)	список
Конструктор	(CONS S-выраж.,список)	список
Предикат	(ATOM S-выражение)	T, либо NIL
Предикат	(EQ атом атом)	T, либо NIL

3.1.1 Функция CAR.

Значением функции является голова списка-аргумента.
Примеры :
$ (CAR '(P Q R))-результат P;
$ (CAR '(A B))- результат A;
$ (CAR '(DOG CAT))-результат DOG;
$ (CAR '((A B) C))-результат (A B);
$ (CAR (A B))-результат не определен, т.к. (A B) рассматривается как функция.

3.1.2 Функция CDR.

Значением функции является хвост списка-аргумента.
Примеры :
$ (CDR '(P Q R))-результат (Q R);
$ (CDR '(B))-результат NIL;

В GNU Common Lisp наряду с простейшими базовыми функциями-селекторами CAR и CDR определены более сложные функции, выделяющие любой из десяти первых элементов :
FIRST - выделяет первый элемент списка, SECOND- второй и т.д.

3.1.3 Функция CONS.

Функция CONS имеет два аргумента - первый S-выражение, вторым аргументом обязательно должен быть список. Назначение функции CONS-конструирование нового списка. Новый список получаем путем добавления первого аргумента к списку- второму аргументу.

Для получения списка, содержащего один элемент, нужно в качестве второго списка указать пустой список NIL.

Примеры :

$ (CONS 'A '(B C))-результат список (A B C);

$ (CONS '(FIRST SECOND)'(THIRD FOURTH FIFTH)) –
результат ((FIRST SECOND) THIRD FOURTH FIFTH);

$ (CONS NIL NIL) - результат пустой список (NIL);

$ (CONS 12 NIL) - результат список (12);

$ (CONS '(A B C) NIL)- список ((A B C));

3.1.4 Связь между функциями CAR,CDR,CONS.

Список, разделенный с помощью CAR и CDR на голову и хвост, можно объединить с помощью функции-конструктора CONS.

Пример.

$ (CAR ‘(FIRST SECOND THIRD)) – результатом будет FIRST;

$ (CDR ‘(FIRST SECOND THIRD)) – результатом будет список (SECOND THIRD);

$ (CONS (CAR ‘(FIRST SECOND THIRD)) (CDR ‘(FIRST SECOND THIRD))) – результатом будет исходный список (FIRST SECOND THIRD).

3.1.5 Предикат ATOM.

Предикат- это логическая функция. Предикат ATOM возвращает значение T (эквивалент TRUE), если его аргументом является атом и NIL(эквивалент FALSE), иначе.

Примеры использования ATOM :

$ (ATOM 'X)-результат T, т.к. аргумент-атом;

$ (ATOM '(1 2 3))-результат NIL, аргумент-список;

$ (ATOM (CDR'(1 2 3)))-результат NIL;

$ (ATOM (CAR'(A B C)))-результат T

3.1.6 Предикат EQUAL.

Предикат EQUAL проверяет эквивалентность любых S-выражений.

Примеры :

$ (EQUAL '(3 3)'(4 3))- NIL, так как списки не совпадают

$ (EQUAL 'Y 'Y)-T, атомы совпадают;

$ (EQL 5.0 5.0)-T

3.1.7 Расширение базовых функций в GNU Common Lisp'e.

Кроме базовых функций в GNU Common Lispe есть целый набор встроенных функций, выполняющих преобразования S-выражений и дополняющих базовые функции.

3.1.7.1 Вложенные вызовы функций CAR и CDR.

Путем использования функций CAR и CDR можно выделить любой элемент списка. Например, для выделения второго элемента второго подсписка нужно записать :

$ (CAR (CDR (CAR (CDR '((A B C) (D E) (F H))))))

В GNU Common Lisp для таких комбинаций CDR и CAR можно использовать более короткую запись в виде одного вызова функции :

(C...R список), где вместо многоточия записывается нужная комбинация букв A (CDR) и D (CDR).НО НЕ БОЛЕЕ 5 БУКВ.

Например, записанная выше конструкция может быть представлена :

(CADADR '((A B C) (D E) (F H)))

Для выделения из списка какого-либо элемента : первого, второго,...,десятого можно использовать специальные функции GNU Common Lisp: FIRST, SECOND, THIRD,FOURTH, FIFTH, SEVENTH, EIGHTH, NINTH, TENTH. Все эти функции имеют один аргумент-список.

Более того, в GNU Common Lisp есть еще две функции, выделяющие элемент из списка:

(NTH n список) - выделяет n-ый элемент из списка-второго аргумента функции NTH;

(LAST список) - возвращает список из одного последнего элемента списка-аргумента.

Примеры использования LAST и NTH :

$ (NTH 4 '(F D C C F H))-выделяет элемент F, т.к. номера элементов начинается с нуля.

$ (LAST '(A B C D E F)) - выделяем (F).

3.1.7.2 Дополнительные предикаты сравнения аргументов.

3.1.7.2.1 Предикат EQ.

Предикат EQ сравнивает два атома, и принимает значение T, если атомы идентичны и NIL в противном случае. EQ сравнивает только атомы и константы T, NIL.

Примеры :

$ (EQ 'X 'Y)-атомы неэквивалентны NIL;

$ (EQ () NIL)-T;

$ (EQ T (ATOM 'CAT))-T

3.1.7.2.2 Предикат "=".

Предикат "=" применяется для сравнения чисел различных типов. Предикат принимает значение T, если значение чисел совпадают вне зависимости от типа, иначе NIL.

Примеры :

$ (= 3 3) – результат T;

$ (= 3 7) - результат NIL

3.1.7.2.3. Предикат NUMBERP.

Предикат NUMBERP имеет один аргумент, который является S-выражением. Предикат принимает значение T, если аргумент является числом любого типа, и NIL, в любом другом случае.

Примеры :

$ (NUMBERP 3.066)-T;

$ (NUMBERP T)-NIL т.к. T-не число.

3.1.7.2.4. Предикат NULL.

NULL имеет один аргумент-список, если список пустой, то NULL принимает значение T, иначе NIL.

3.1.7.2.5. Функция LIST.

Эта функция может иметь любое количество аргументов, эта функция формирует список из аргументов.

Примеры :

$ (LIST '1 '2) - реэультат (1 2);

$ (LIST 'A 'B 'C 'D (+ 3 4)) - результат (A B C D 7).

4. Объявление функций.

В функциональных языках программирования различают описания именованных и неименованных функций. Для описания неименованных функций в Лиспе используется конструкция LAMBDA.

4.1 Неименованные функции.

Неименованные функции используются для выполнения однократных преобразований или вычислений, такие действия носят локальный характер и используются только одной функцией. Наиболее часто неименованные функции используются в функционалах и макросах.

4.1.1 Описание неименованных функций в GNU Common Lisp'e.

Для описания неименованных функций в GNU Common Lisp'e используются lambda-вызовы, имеющие следующий вид :

((lambda (<список формальных параметров>)

< тело функции>)

< список фактических параметров>)

Пример. Для функции f1(x,y), которая возвращает в качестве результата y в случае атомарного x и список ‘(x,y) – в противном случае, примером lambda-вызова может послужить :

```
((lambda (x y)
    ((atom (car x)) y)
    (cons x y))
(car '((f) g h))'(r t y))
```

4.2. Описание именованных функций.

4.2.1 Описание именованных функций в GNU Common Lisp'e.

Для объявления именованных функций используются конструкции :

(DEFUN <имя функции> (список формальных параметров)

<lambda -вызов>)
(DEFUN <имя функции> (список формальных параметров)
<тело функции>)

DEFUN - служебное слово (DEfine FUNction - определение функции), которое обязательно должно начинать описание функции. Имя функции может быть любым символьным атомом, рекомендуется давать такие имена функциям, которые отражают смысловое содержание функции. Список формальных параметров представляет собой список символьных атомов, записанных через пробел. Тип параметров не указывается. Тело функции может содержать либо ветвления, либо представлять собой суперпозицию вызовов функций.

Результат, полученный при выполнении функций, связывается с именем функции.

Для примера рассмотрим описание и вызов функции, которая из списка выделяет второй и четвертый элементы и конструирует из них список.

Задачу конструирования нового списка из элементов старого можно решить различными способами, используя базовые функции CONS, CDR, CAR и функции GNU Common Lisp: SECOND, FOURTH, LIST.

Описание функции непосредственно в интерпретаторе будет выглядеть следующим образом :

```
$ (DEFUN SFLIST (LST) (CONS (SECOND LST) (CONS (FOURTH LST) NIL)))
```

При использовании lambda-вызова описание имеет вид :

```
$ (DEFUN SFL (LST)
((lambda (X Y)
(CONS X(CONS Y NIL)))
(SECOND LST)(FOURTH LST)))
```

Эта же функция может быть описана с использованием других функций GNU Common Lisp: LIST, CADR, CADDDR :

```
$ (DEFUN SFILST1 (LST)
(LIST (CADR LST)(CADDDR LST)))
```

5. Вызов функции.

Вызов функции записывается следующим образом :

(имя функции <список фактических параметров>)

Например, вызов функции SFLIST :

```
$ (SFLIST '(DOG CAT COW PIG))
```

дает в качестве результата список (CAT PIG).

7. Задание на лабораторную работу.

7.1. Ознакомиться с описанием лабораторной работы.

7.2. Выполнить примеры.

7.3. Выполнить свой вариант задания, вариант выдает преподаватель. Задание выполнить различными способами, применяя простейшие и функции из расширения базовых функций GNU Common Lisp (newLISP-tk).

Задание 1.

Описать неименованную функцию для объединения голов трех списков в один список, исходные данные взять из ***таблицы 4***.

Задание 2.

Описать именованную функцию для создания нового списка из элементов нескольких исходных списков. В качестве исходных списков использовать списки ***таблицы 4***. Номера элементов списков взять в ***таблице 5***.

Таблица 4. Исходные списки.

Вариант	Исходные списки		
	1	2	3
1	(Y U I)	(G1 G2 G3)	(KK LL MM JJJ)
2	(G55 G66 G777)	(9 (F G) I)	(N I L T D J (II JJ))
3	((PI) V (H J K))	(R YU (H KJ KL)	(U II OO LL PP (3 4 5))
4	(T (U U1 U2) (U4 U6 U8))	(4 6 (7 8 9))	(78 89 90 67 45)
5	(9 (() 8 88 888))	(H (J K L) (UJN))	(C B (N M I) (T Y U))
6	(T Y D E F (NL KM LM) JL)	(+ 2 3)	(*(+ 6 8) (- 70 8))
7	(TYPE PRINT DEL)	(H (H J O) (UJ N))	(READ SAVE LOAD (TXT))
8	(GOAL FUNCTOR CLAUSE (DATA BASE))	(2 5(5 4 6)8)	(L (K (K I)U))
9	(DOG (CAT) FOX ())	(RET GET PUT OUT IN)	(MOV ADD (MUL DEV))
10	(FIR SED (1 2 3) (5) ())	(H J U K (L M N) (D E L))	(4 5(6 7))
11	(PRIM SD FLAG () (GHG))	(1 56 98 52)	(T 2 3 4 Y H)
12	(H G (U J) (T R))	(2 1 (+ 4 5))	(TYPE CHAR REAL (H G))
13	(REM FFG HHJ (H)J G D)	(2 34 56 78 (7 8))	(UN Y LOOP)
14	(T (HJ (JH KL)) K)	(67 54 (8 9 0)(4 6))	(K F G H)
15	(3 (3 4 5 Y U)((T Y))	(G H (6 7 8) 8 9 0 7 6)	((5 T 7 Y H) U)
16	(Z X C S A D F)	((R)(30)(3) 23))	(U I 8 9 6 5 4 3 (1 2 3))
17	(V B N J H)	((Y U I)(H J K) (8) 78)	(df FG HJ K L (O 0 9))
18	(D F G H J K)	(1 2 3 4 5 6 (4 5) 4)	(ER RT TY 5 6 6 5)
19	(H G (2 3) 8 7 (T R))	(2 1(+ 4 5))	(TY PE CH AR RE AL (H G))
20	(RM F G H J (J G D))	(2 3 4 5 6 (7 8))	(U N Y L O O P)
21	(T HJ JH K L (K)	(6 7 5 4 (8 9 0)(4 6))	(K 2 T F G H)
22	(3 3 4 5 Y U)	(G (6 8) 8 9 7 6)	((5) T () 7 Y H U)
23	(Z 2 A D F)	((R) (30) 3 4 23)	(U I 8 9 6 5 4 3 (1 2 3))
24	(V B N J H)	(Y U I H (J K) (8) 78)	(df F K L (O 0 9))
25	(DF GH JK)	(1 2 5 6 (4 5) 4)	(ER RT TY 5 6 6 5)

Таблица 5. Номера элементов.

Вариант	Список 1	Список 2	Список 3
1	2	2	3
2	3	2	6
3	1	3	6
4	2	3	4
5	2	3	3
6	6	2	2
7	3	2	3
8	4	3	2
9	3	4	3
10	4	4	3
11	5	3	2
12	3	3	3
13	4	4	2
14	2	3	3
15	2	3	2
16	5	4	7
17	3	4	6
18	5	3	2
19	3	3	3
20	4	6	2
21	2	6	3
22	2	3	2
23	5	4	7
24	3	4	6
25	6	6	6

Задание 3.

Описать именованную функцию в соответствии с вариантом индивидуального задания в Таблице 6.

Таблица 6. Варианты индивидуального задания.

Вариант.	Задача.
1.	Есть два списка. Если первый элемент списка есть натуральное число, то возвратить второй список, иначе возвратить список из головы второго и хвоста первого.
2.	Написать функцию ,которая возвращает квадратный корень из аргумента, если аргумент является числом, последний элемент аргумента-списка, если аргумент – список, аргумент – иначе.
3.	Есть пять чисел. Написать функцию , формирующую список из максимального и минимального по модулю чисел, если минимальное и максимальное числа – целые, среднее арифметическое минимального и максимального чисел - иначе.
4.	Написать функцию, которая для аргумента-числа проверяет, является ли оно степенью двойки.
5.	Написать функцию, которая для заданных координат X1,Y1 и X2,Y2 возвращает расстояние между ними. Координаты могут иметь отрицательное значение.
6.	Написать функцию, которая по заданному вещественному числу формирует список из трех элементов. Первый элемент – знак числа, второй – модуль числа, третий – ближайшее к нему целое число.
7.	Написать функцию, которая для трех аргументов-чисел проверяет, является ли третье число результатом возведения в степень первого числа с показателем, равным второму числу.
8.	Есть список. Найти сумму первого, третьего и седьмого элементов списка, если указанные элементы – числа, возвратить последний элемент списка – иначе.
9.	Написать функцию вычисления дискриминанта квадратного уравнения.
10.	Написать функцию, которая для аргумента-списка формирует список-результат по правилу : если первый и последний элементы списка-аргумента – четные положительные целые числа, то включить в список-результат первым элементом – квадрат последнего элемента исходного списка, вторым – четвертую степень первого; в противном случае сформировать список из первого и последнего элементов.
11.	Написать функцию, которая для аргумента-списка формирует список-результат по правилу : если первый и последний элементы списка-аргумента –символы, то сформировать список из первого и последнего элементов, в противном случае возвратить исходный список, из которого удален второй элемент.
12.	Есть два списка. Написать функцию формирующую список из трех подсписков. Первый подсписок содержит голову первого списка и третий элемент второго. Второй подсписок содержит второй элемент второго списка и последний элемент первого.
13.	Есть списки, к примеру, '(1 2 3 4 5) '(7 6 5 7). Если произведение первых элементов исходных списков есть положительное число, то объединить в результирующий список последние элементы. В противном случае построить список из последнего элемента первого списка и хвоста второго.
14.	Есть три числа. Построить список из кубов этих чисел, если все три числа – нечетные, возвратить сумму чисел – иначе.
15.	Есть список и некоторый объект. Написать функцию, возвращающую новый список, в котором объект замещает первый элемент списка, если первый элемент списка и объект являются атомами, последний элемент списка - иначе.

16.	Дан список чисел. Написать функцию, возвращающую в случае первого четного элемента исходный список, в котором первые три числа возведены в квадрат, иначе - исходный список, в котором первые три числа возведены в куб.

17.	Даны два произвольных лисповских объекта. Написать функцию, которая возвращает список из суммы и произведения объектов, если оба объекта – числа, список из результатов проверки объектов на атомарность – иначе.
18.	Дан список. Написать функцию, которая в случае атомарности первого и последнего элементов списка возвращает список из указанных элементов, результат проверки третьего элемента на принадлежность к списочным объектам – иначе.
19.	Дан список lst. Написать функцию, которая для случая, когда первый элемент lst является списком, возвращает список из последнего элемента lst в качестве головы и первого элемента lst в качестве хвоста, список lst, из которого удален второй элемент – иначе.
20.	Написать функцию, которая возвращает квадрат аргумента-четного числа, натуральный логарифм аргумента-нечетного числа и результат проверки аргумента на принадлежность к списочному типу – иначе.
21.	Есть список lst, описывающий вызов арифметической функции. Написать функцию, которая а случае четности результата вычисления lst производит его проверку на положительность, а в противном случае выдает сам lst. Вычисление lst производить с помощью встроенной функции eval.

22.	Есть две лисповские формы : expr1 и expr2. Написать функцию, которая сравнивает результаты вычисления форм и в случае эквивалентности результатов вычисления формирует список из этих форм. В противном случае формируется список из результатов вычисления форм. Вычисление форм производить с помощью встроенной функции eval.
23.	Написать функцию, которая по заданному целому числу формирует список двух элементов. Первый элемент списка - это символьный атом, обозначающий знак числа, второй элемент – остаток от деления числа на 2.
24.	Написать функцию, которая для заданных списков lst1 и lst2 возвращает список, содержащий их первые и последние элементы. Порядок следования элементов в результирующем списке определяется вторым элементом lst2 : если это число, то вначале идут первый и последний элементы lst1, иначе – первый и последний элементы lst2.
25.	Написать функцию, которая по двум числам формирует список из трех элементов. Первый элемент – это результат целочисленного деления чисел, второй есть результат умножения чисел, третий элемент есть символьный атом, обозначающий знак числа : плюс, минус, ноль. Функция должна содержать проверку делителя на ноль !

Все результаты должны быть обоснованы : почему получен именно такой результат.

8. Содержание отчета по лабораторной работе.

Отчет по лабораторной работе должен содержать :

− формулировку цели и задач;

− результаты выполнения заданий по пунктам, обоснование полученных результатов и выбранных структур функций;

- скриншоты выполненных заданий;

− выводы по проведенным экспериментам.

К отчету в формате doc или odt прикладываются ЛИСП-файлы в формате *.lsp.

Литература.

1. Хювенен Э., Сеппянен Й. Мир Лиспа. В 2-х т. Пер. с финск. – М.: Мир, 1990.
2. GCL (GNU Common Lisp) Online Manual // http://cubist.cs.washington.edu/~tanimoto/gcl-si.html
3. Gnu Common Lisp for Microsoft Windows // https://www.cs.utexas.edu/users/novak/gclwin.html
4. Введение в Common Lisp: Установка и первый запуск // http://devhead.ru/read/vvedenie-v-common-lisp-ustanovka-i-pervyj-zapusk