МЕТОДИ ТА СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ

ЛАБОРАТОРНА РОБОТА № 2

Опис і виклик функцій в мові Ліспи.

1. **Мета і задачі.** Метою роботи є вивчення базових функцій організації та обробки списків , а також методи опису та вибору нерекурсивних функцій у мові програмування Ліспи (на прикладі одного з відомих діалогів мови Ліспи).

Основні завдання:

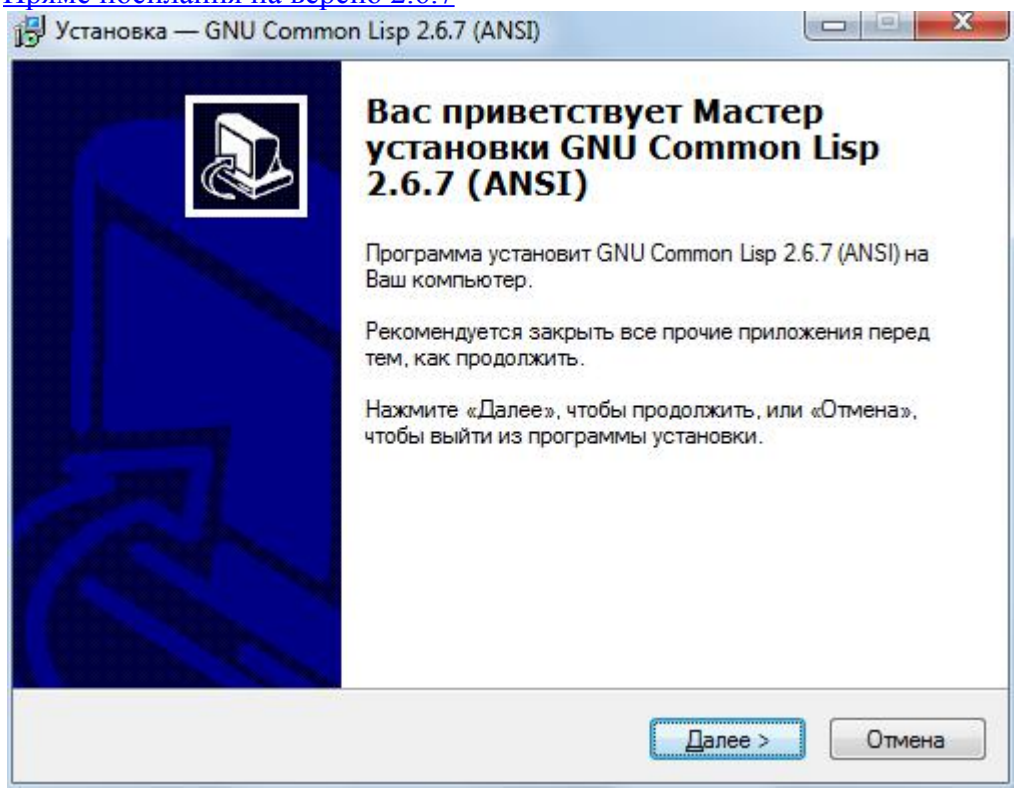
- отримати навички роботи з інтерпретатором Ліспа для вибранного діалекту .
- вивчити роботу примітивних базових функцій списочного складання .
- визначити роботу базових функцій з розширення набору примітивних функцій та їх введення до примітивних базових функцій .
- ознайомлення з описом неіменованих функцій у Ліспі .
- вивчення прийому опису іменованих функцій через неіменовані та із застосуванням сучасної скороченої нотації .

2. Робота з інтерпретатором Лиспа . Є можливість виконувати лабораторні практики в середовищі GCL (GNU Common Lisp).

2.1 Робота в середовищі Microsoft GNU Common Lisp.

Версію для Windows можно скачати за цією адресою

Пряме посилання на версію 2.6.7



Приклади :

`$(* 4 5)` - виклик функції множення для двох аргументів;

`$(* 5 6 7 8)` — те ж саме для чотирьох аргументів;

$(- 12 3) - виклик функції віднімання **12-3**

Для виходу з середовища Microsoft GNU Common Lisp необхідно набрати в командному рядку інтерпретатора :

$ (QUIT).

2.1.1 Рекомендована послідовність дій під час роботи з інтерпретатором.

- розробити текст програми ;
- записати розроблений текст у файл із розширенням **.lsp**.
- викликати інтерпретатор із зазначенням виконуваного файлу .

Для скорочення обсягу програм у першій лабораторній роботі використовується використання псевдофункції , зв'язуючої імені та значення **(SETQ ім'я значення)**. До прикладу , потрібно виконувати багатократну обробку списку **'(acd (ef))**. Обозначим цей список **l1** і зробимо це так **(SETQ l1 '(acd (ef)))**. Після такого зв'язування там , де використовується значення списку, можна використовувати ім'я **l1**.

2.1.2 Рекомендована структура програм.

- зв'язування імен та значень ;
- виклик функції , або опис функцій , тобто власне текст програми ;
- функція переключення вхідного потоку (RDS)

3. Базові функції Ліспа.

Примітивні (простіші) базові функції обробки можна порівняти з основними арифметичними діями : вони прості та їх мало.

Призначение цих функцій:

- розщеплення S-виразів (селектори);
- складання S-виразів (конструктори);
- аналіз S-виразів (предікати).

3.1 Реализація базових функцій в GNU Common Lisp'e.

Таблиця 1. Примітивні базові функції.

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 Функція CAR.

Результатом виконання функції є голова списку-аргумента.

Приклади :

$ (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 визначений, оскільки **(A B)** розглядається як функція.

3.1.2 Функція CDR.

Результатом виконання функції є хвіст списку-аргумента.

Приклади :

$ (CDR '(P Q R))-результат **(Q R)**;

$ (CDR '(B))-результат **NIL**;

У GNU Common Lisp наперед з простейшими базовими функціями - селекторами **CAR** та **CDR** визначені більш складні функції , що виділяють будь-якого з десяти перших елементів :
FIRST - виділяє перший елемент списку, **SECOND** - другий і т.д.

3.1.3 Функція CONS.

Функція **CONS** має два аргументи — перший S-вираз, другим аргументом обов'язково повинен бути список. Призначення функції **CONS** - конструювання нового списку. Новий список отримуємо шляхом додавання першого аргумента до списку - другому аргументу — в якості першого елемента. Для отримання списку , що містить один елемент , потрібно в якості другого аргумента вказати пустий список NIL.

Приклади :

$ (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 Св'язок між функціями CAR,CDR,CONS.

Список, розщеплений за допомогою **CAR** та **CDR** на голову і хвіст, можна об'єднати за допомогою функції-конструктора **CONS**.

Приклад.

$ (CAR '(FIRST SECOND THIRD)) – результатом буде **FIRST**;

$ (CDR '(FIRST SECOND THIRD)) – результатом буде список **(SECOND THIRD)**;

$ (CONS (CAR '(FIRST SECOND THIRD)) (CDR '(FIRST SECOND THIRD))) – результатом буде вхідний список **(FIRST SECOND THIRD)**.

3.1.5 Предікат ATOM.

Предікат - это логічна функція, яка приймає значення істина або брехня. Предикат **ATOM** повертає у якості результату константу **T** (еквівалент TRUE), якщо його аргументом є атом і **NIL** (еквівалент **FALSE**), інакше. Приклади використання **ATOM** :

$ (ATOM 'X) - результат **T**, оскільки 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.

Предікат **EQUAL** перевіряє еквівалентність будь-яких S-виразів.

Приклади :

$ (EQUAL '(3 3)'(4 3)) - **NIL**, оскільки списки не співпадають

$ (EQUAL 'Y 'Y) - **T**, атоми співпадають;

$ (EQL 5.0 5.0) - **T**

3.1.7 Розширення базових функцій в GNU Common Lisp'e.

Крім базових функцій в GNU Common Lispe є цілий набір вбудованих функцій, що виконують перетворення S-виразів та доповнюючих базові функції.

3.1.7.1 Вкладені визови функцій CAR та CDR.

Шляхом використання функций **CAR** и **CDR** можно отримати будь-який елемент списку. Наприклад, для отримання другого елемента другого підсписку треба записати :

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

В GNU Common Lisp для таких комбінацій **CDR** и **CAR** можна використовувати більш стислу нотацію у вигляді одного визова функції :

(C...R список), де замість багатокрапки записується потрібна комбінація букв **A (CDR)** и **D (CDR)**. АЛЕ НЕ БІЛЬШЕ 5 БУКВ.

Наприклад, записана вище конструкція може бути представлена :

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

Для отримання зі списку будь-якого елементу : першого, другого,...,десятого можна використовувати спеціальні функції GNU Common Lisp: **FIRST**, **SECOND**, **THIRD**, **FOURTH**, **FIFTH**, **SEVENTH**, **EIGHTH**, **NINTH**, **TENTH**. Всі ці функції мають один аргумент-список.

Більш того, в GNU Common Lisp є ще дві функції, відокремлюючі елемент зі списку:
(NTH n список) - виокремлює n-ий елемент зі списку - второго аргумента функції NTH;
(LAST список) — повертає список з одного останнього елемента списку-аргумента.
Приклади використання **LAST** и **NTH** :
$ (NTH 4 '(F D C C F H)) - виокремлює елемент **F**, оскільки номера елементов починаються з нуля.
$ (LAST '(A B C D E F)) - результат **(F)**.

3.1.7.2 Додаткові предикати порівняння аргументів.
3.1.7.2.1 Предікат EQ.

Предікат **EQ** порівнює два атоми, і приймає значення **T**, если атоми ідентичні та **NIL** в протилежному випадку. **EQ** порівнює тільки атоми і константи **T**, **NIL**. Приклади :

$ (EQ 'X 'Y) - атоми нееквівалентні **NIL**;
$ (EQ () NIL) - **T**;
$ (EQ T (ATOM 'CAT)) - **T**
3.1.7.2.2 Предікат "=".

Предікат "**=**" застосовується для порівняння чисел різних типів. Предікат приймає значення **T**, якщо значення чисел співпадають поза залежністю від типа, инакше **NIL**. Приклади :

$ (= 3 3) – результат **T**;
$ (= 3 7) - результат **NIL**
3.1.7.2.3. Предікат NUMBERP.

Предікат **NUMBERP** має один аргумент, який є S-виразом. Предікат приймає значення **T**, якщо аргумент є числом довільного типу, та **NIL**, в протилежному випадку. Приклади :

$ (NUMBERP 3.066) - **T**;
$ (NUMBERP T) - **NIL** оскільки **T** - не число.
3.1.7.2.4. Предікат NULL.

NULL має один аргумент-список, якщо список пустий, то **NULL** приймає значення **T**, інакше **NIL**.
3.1.7.2.5. Функція LIST.

Ця функція може мати довільну кількість аргументів, вона формує список з аргументів.
Приклади :
$ (LIST '1 '2) - реэультат **(1 2)**;
$ (LIST 'A 'B 'C 'D (+ 3 4)) - результат **(A B C D 7)**.

4. **Декларація (визначення) функцій.**

У функціональних мовах програмування відрізняються описи іменованих та неіменованих функцій . Для опису неіменованих функцій у Ліспе використовується конструкція **LAMBDA**.
4.1 Неіменовані функції.

Неіменовані функції використовуються для виконання разових перетворень або обчислень, такі дії мають локальний характер і використовуються лише однією функцією. Найбільш часто неіменовані функції використовуються в функціоналах і макросах.
4.1.1 Опис неіменованих функцій в GNU Common Lisp'e.

Для опису нейменованих функцій в GNU Common Lisp'e використовуються **lambda**-виклики, мають наступний вигляд :

```
((lambda (<список формальних
    параметрів>) < тіло
    функції> )
    < список фактичних параметрів> )
```

Приклад. Для функції **f1(x,y)**, яка повертає у якості результату **y** у випадку атомарного **x** та список **'(x,y)** – в протилежному випадку, прикладом **lambda**-виклика може розглядатися :

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

4.2. Опис іменованих функцій.

4.2.1 Опис іменованих функцій в GNU Common Lisp'e.

Для декларації іменованих функцій використовуються конструкції :

```
(DEFUN <ім'я функції> (список формальних параметрів)
    <lambda-виклик> )
(DEFUN <ім'я функції> (список формальних параметрів)
    <тело функції> )
```

DEFUN - служебове слово (DEfine FUNction - визначення функції), яке обов'язково повинно починати опис функції. Ім'я функції може бути будь-яким символьним атомом, рекомендується давати такі імена функціям, які відображують сенс змісту функції. Список формальних параметрів є список символьних атомів, записаних через пробіл. Тип параметрів не вказується. Тіло функції може мати або розгалудження, або представляти суперпозицію викликів функцій.

Результат, отриманий при виконанні функцій, пов'язується с іменем функції.

Для прикладу розглянемо опис та виклик функції, яка зі списку виокремлює другий і четвертий елементи і конструює з них список.

Задачу конструювання нового списка з елементів старого можно виконати декількома способами, використовуючи базові функції **CONS**, **CDR**, **CAR** та функції 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. **Виклик функції.**

Виклик функції записується наступним чином :

```
(имя функции <список фактических параметров>)
```

Наприклад, виклик функції **SFLIST** :

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

дає в результаті список **(CAT PIG)**.

7. **Завдання на лабораторну роботу.**

7.1. Ознайомитися з описм лабораторной работы.

7.2. Виконати приведені приклади.

7.3. Виполнить свій варіант завданняг. Завдання викрнати різними способами, застосовуючі найпростіші функції з розширення базових функцій GNU Common Lisp.

Завдання 1.
Описати нейменовану функцію для об'єднання голів трьох списків в один список, вхідні дані взяти з ***таблиці 4***.

Завдання 2.
Описати іменовану функцію для створення нового списка з елементів декількохх вхідних списків. В якості вхідних списків використовувати списки ***таблиці 4***. Номера елементів списків взяти в ***таблиці 5***.

Таблиця 4. Вхідні 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.

Описати іменовану функцію у відповідності з варіантом індивідуального завдання в Таблиці 6.

Таблиця 6. Варіанти індивідуального завдання.

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 !

Усе результати повинні бути обґрунтовані : чому отриманий саме такий результат.

8. Зміст звіту з лабораторної работи №2.

Звіт з лабораторної роботи повинен мати :

− формулювання мети і завдань;

− результати виконання завдань по пунктах, обґрунтування отриманих результатів і обраних структур функцій;

− скріншоти виконаних завдань і прикладів обчислень;

− висновки за проведеними експериментами.

До звіту в форматі doc або odt прикладываются ЛІСП-файли в форматі *.lsp.

Література.

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