ЛАБОРАТОРНАЯ РАБОТА 1

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, а также с применением современной сокращенной нотации.

2. Работа с интерпретатором Лиспа.

Предусматривается возможность выполнения лабораторного практикума в средах CLISP, Microsoft muLISP, newLISP-tk и newLISP-GS.

2.1 Работа в среде Microsoft muLISP.

Исполняемым файлом интерпретатора muLISP является mulisp.com (mulisp_2.com – для русифицированной версии). Русифицированную версию интерпретатора рекомендуется использовать в комплекте с программой-русификатором для MS DOS, например, keyrus.com или rk.com.

Для работы с интерпретатором необходимо после запуска mulisp.com и появления приглашения $ набирать либо описание функции, либо ее вызов.

Примеры:

$(* 4 5) – вызов функции умножения для двух аргументов;

$(* 5 6 7 8) – то же самое для четырех аргументов;

$(− 12 3) – вызов функции вычитания 12−3.

Для выхода из среды Microsoft muLISP необходимо набрать в командной строке интерпретатора:

$(SYSTEM).

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 SETQ, 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 (непосредственно текст Лисп-программы);
- функция переключения входного потока (RDS).

2.2 Работа в среде newLISP-tk.

Исполняемым файлом для запуска интегрированной среды newLISP-tk в Windows является newlisp-tk.exe (по умолчанию устанавливается в каталог C:\Program Files\newlisp).

Для работы с интерпретатором необходимо после запуска интегрированной среды либо набирать в командной строке описание функции, либо загрузить на исполнение файл Лисп-программы *.lsp : FILE | LOAD.... Файл может быть также загружен из командной строки: >(load “имя_файла.lsp”). Имена файлов и директорий начинаться с латинской буквы и не должны содержать символов кириллицы.

Для выхода из среды newLISP-tk необходимо набрать в командной строке интерпретатора:
>(EXIT),
либо выбрать в главном меню FILE | EXIT

2.2.1 Рекомендуемая последовательность действий при работе с интерпретатором.

При разработке программы допускается использовать последовательность действий, аналогичную изложенной в п. 2.1.1 для muLISP’a, а именно:

- разработать текст программы;
- записать разработанный текст в файл с расширением .lsp;
- запустить интерпретатор;
- загрузить файл программы.

Кроме того, Файл программы может быть создан непосредственно в среде newLISP-tk, более подробно см. пункт 6.

2.3 Работа в среде newLISP-GS.

В Windows исполняемым файлом интерпретатора является newlisp.exe. По умолчанию он устанавливается в каталог C:\Program Files\newlispGS. Интегрированная среда разработки (англ. integrated development environment, IDE) в newLISP-GS включает графический редактор, который написан непосредственно на Лиспе (файл newlisp-edit.lsp) и при запуске загружает guiserver.lsp – модуль организации интерфейса с guiserver.jar – серверным Java-приложением для генерации графических пользовательских интерфейсов.

Для вызова интегрированной среды при установке в Windows создаётся специальный ярлык, в указании на объект которого прописано следующее:

“C:\Program Files\newlispGS\guiserver.jar” 47011 newlisp-edit.lsp /local/newLISPsplash.png.

Здесь 47011 – номер порта, используемого guiserver.jar.

В состав интегрированной среды входит редактор с подсветкой пар скобок и ключевых слов языка, а также консоль вывода с возможностью работы в интерактивном режиме. Входящие в состав пользовательского интерфейса кнопки имеют всплывающие подсказки. Посредством нажатия кнопки «Run editor content» набранный в редакторе текст программы запускается для исполнения интерпретатором.

Интерпретатор языка newLISP написан на «чистом» C, что делает и язык, и его среду переносимыми на Windows, Linux, MacOS, OS/2. Более подробно об установке newLISP-GS в операционной системе Ubuntu, созданной на основе Debian GNU/Linux см. п. 7.

2.4 Работа в среде CLISP.

В Windows исполняемым файлом интерпретатора является clisp.exe. Путь к CLISP в Windows 10: C:\Program Files (x86)\clisp-2.49 (для версии 2.49). CLISP написан на «чистом» C и Common Lisp [6,7]. CLISP является кроссплатформенным, запускается на всех Unix-подобных операционных системах. На рис. 1 представлено окно среды CLISP, пример загрузки файла с исходным текстом программы и вызов функции my_member из описанных в этом файле (операционная система Ubuntu 16.04 LTS). Для выхода из среды CLISP необходимо набрать в командной строке интерпретатора (quit) либо (exit).

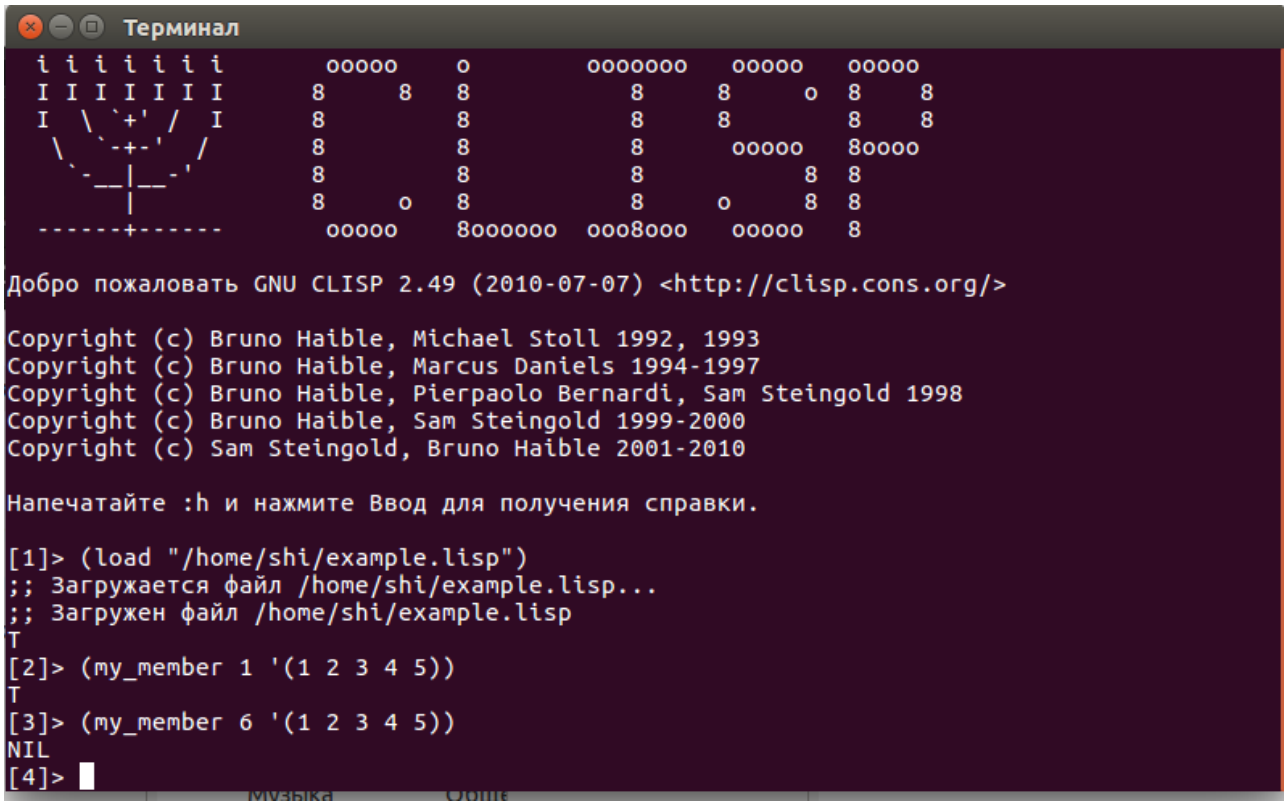


Рис. 1. Среда CLISP

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 (функции-селекторы);
- составлении S-выражений (функции-конструкторы);
- анализ S-выражений (функции-предикаты).

3.1 Реализация базовых функций в Common Lisp и muLISP.

Таблица 1. Примитивные базовые функции.

Назначение.	Вызов.	Результат.
Селектор	(CAR *список*)	S-выражение
Селектор	(CDR *список*)	список
Конструктор	(CONS *S-выражение список*)	список
Предикат	(ATOM *S-выражение*)	T, либо NIL
Предикат	(EQ *атом1 атом2*)	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)) – результатом будет список (Q R);
$ (CDR ‘(B)) – результат NIL.

В Common Lisp и muLISP наряду с простейшими базовыми функциями-селекторами CAR и CDR определены более сложные функции, выделяющие любой из десяти первых элементов, а именно: 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 в качестве головы к списку-второму аргументу.

Для получения списка, содержащего один элемент, нужно в качестве второго списка указать пустой список (либо NIL, либо явным образом задав ‘()).

Примеры:

$ (CONS ‘A ‘(B C)) – результатом будет список (A B C);
$ (CONS ‘(FIRST SECOND) ’(THIRD FOURTH FIFTH)) даст результат ((FIRST SECOND) THIRD FOURTH FIFTH);
$ (CONS NIL NIL) – результатом будет список (NIL);
$ (CONS 12 NIL) – результат есть одноэлементный 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. К примеру, 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)) – результат NIL, так как списки не совпадают;
$ (EQUAL ‘Y ‘Y) – результат T, поскольку атомы совпадают;
$ (EQL 5.0 5.0) – результат T.

3.1.7 Расширение базовых функций в Common Lisp и muLISP.

Кроме базовых функций в данных реализациях Лиспа 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 рассматриваемых диалектах Лиспа 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 пяти букв!*

Например, записанную выше конструкцию можно представить следующим образом:

(CADADR ‘((A B C) (D E) (F H)))

Для выделения из списка элементов с 1-го по 10-й включительно можно использовать специальные функции: FIRST, SECOND, THIRD, FOURTH, FIFTH, SEVENTH, EIGHTH, NINTH, TENTH. Все эти функции имеют один аргумент-список.

Также в Common Lisp и muLISP есть еще две функции, выделяющие элемент из списка:

- (NTH n список) – выделяет *n*-й (*начиная с нуля*) 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)) – выделяется элемент F, т.к. номера элементов начинается с нуля;
$ (LAST ‘(A B C D E F)) – выделяется список (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, результат – NIL;
$ (EQ () NIL) – результат T;
$ (EQ T (ATOM ‘CAT)) – результат 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) – результат T;
$ (NUMBERP T) – результат 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) – результат (1 2);
$ (LIST ‘A ‘B ‘C ‘D (+ 3 4)) – результат (A B C D 7).

3.2 Особенности реализации базовых функций в newLISP.

Таблица 2. Соответствие базовых функций в newLISP базовым функциям Common Lisp и muLISP.

Common Lisp и muLISP	newLISP
car	first
cdr	rest
cons	cons
atom	atom?
equal	=

Таблица 3. Реализация расширения базовых функций Common Lisp/muLISP в newLISP.

Common Lisp и muLISP	newLISP
nth	nth
last	last[1]
null	null?
list	list

4. Объявление функций.

В функциональных языках программирования различают описания именованных и неименованных функций. Для описания неименованных функций в Лиспе используются две конструкции: LAMBDA и NLAMBDA[2].

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 Common Lisp и muLISP.

Для описания неименованных функций в указанных диалектах Лиспа 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))
```

Примечание. В отличие от muLISP, в Common Lisp отсутствует так называемый «встроенный COND», существенно сокращающей тексты описаний функций. Поэтому условные предложения (COND, IF-ELSE) в программе на Common Lisp прописываются явно. Более подробно по синтаксису программ на Common Lisp см. [6,7].

4.1.2 Описание неименованных функций в newLISP.

В общих чертах синтаксис описания неименованных функций в newLISP сходен с описанием аналогичных конструкций в Common Lisp и muLISP. Как и в Common Lisp, объявление управляющих структур if и cond в случае наличия разветвлений в структуре преобразования (вычисления) является обязательным.

Так, lambda-вызов из примера в п. 4.1.1 будет выглядеть следующим образом:

1 В newLISP функция last возвращает последний элемент списка-аргумента
2 В отличие от LAMBDA NLAMBDA не вычисляет фактических параметров и применяется не во всех реализациях языка Лисп.

((lambda (x y) (if (atom? (first x)) y) (cons x y)) (first ‘((f) g h)) ‘(r t y))

4.2. Описание именованных функций.

4.2.1 Описание именованных функций в Common Lisp и muLISP.

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 (англ. 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 заданного исходного в Common Lisp и muLISP можно решить различными способами, используя базовые функции CONS, CDR, CAR и функции из расширенного набора: SECOND, FOURTH, LIST и т.п.

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 Common Lisp: LIST, CADR, CADDDR:

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

4.2.2 Описание именованных функций в newLISP.

В общих чертах синтаксис именованной функции имеет следующий вид:

(define (<*имя функции*> <*список формальных параметров*>) <*тело функции*>)

Так, описание и вызов функции из примера в п.4.2.1 можно задать следующим образом:

```
(define (sfl lst) ((lambda (X Y) (cons X (cons Y ‘()))) (nth 1 lst) (nth 3 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).

6. Работа с редактором и особенности написания и отладки программы в среде newLISP-tk.

Основывается на понятии контекста или пространства имен. Контекст соответствует списку символов, с которыми для заданного круга задач связывается значение, определение функции и другие свойства. Данный список называется также списком объектов. Причем в программе в разных списках объектов могут содержаться символы с одинаковыми именами, но имеющие разные значения. Примеры: методы со сходным функциональным назначением, реализация которых зависит от объекта.

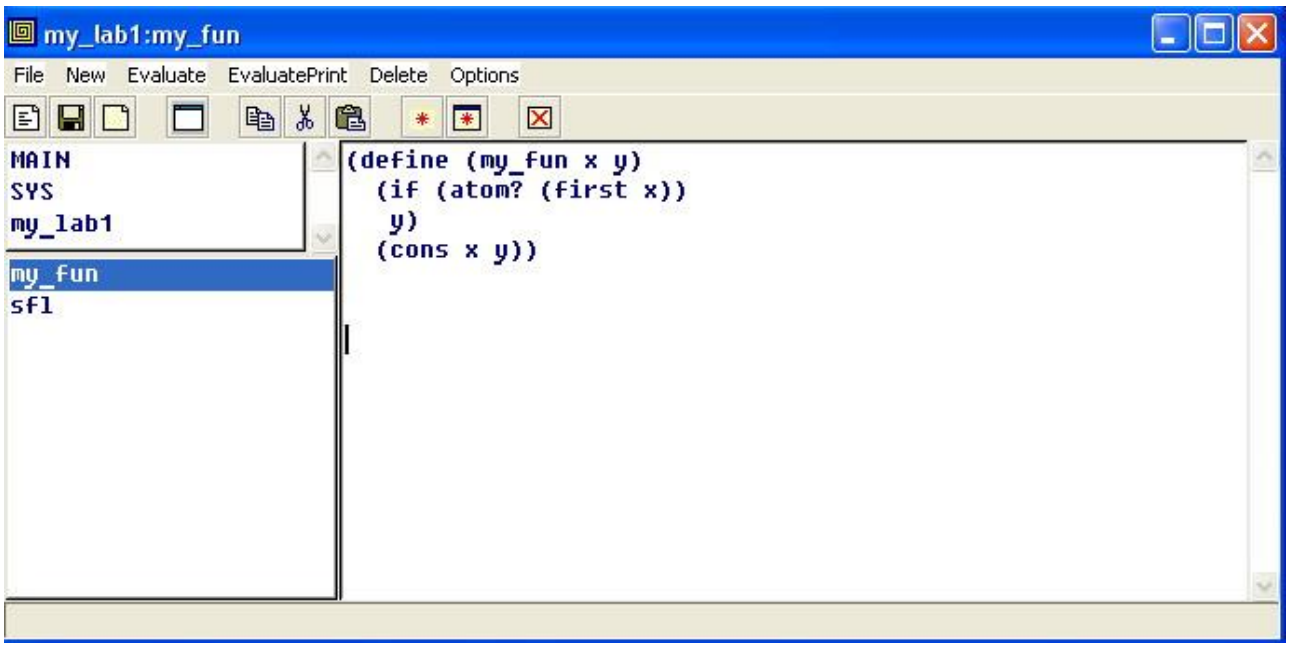


Рис. 2. Редактор интегрированной среды newLISP-tk

Последовательность действий по созданию программы непосредственно в среде интерпретатора состоит в следующем:

1) Запустить интегрированную среду newLISP-tk.
2) Запустить редактор, для чего в главном меню интегрированной среды выбрать Edit.
3) В главном меню редактора (рис. 2) выбрать File | Create Context.
4) В появившемся окне ввести имя контекста, например, my_lab1. Имя нового контекста при этом должно появиться в левом верхнем окне, рис. 2.
5) Набрать описание функции в окне редактирования (правое на рис. 2).
6) В главном меню редактора выбрать Evaluate. При корректности синтаксиса набранной функции ее имя должно появиться в левом нижнем окне (рис. 2), которое содержит список описанных пользователем функций в текущем контексте.
7) Повторять шаги 5–6 до тех пор, пока не будет введено описание всех функций программы.

8) В главном меню редактора выбрать File | Save Context As. В появившемся окне выбрать директорию и имя файла для сохранения.
9) Для запуска функций программы необходимо в окне редактирования набрать в соответствии с принятым в Лиспе синтаксисом имя функции с фактическими параметрами и выбрать EvaluatePrint в главном меню редактора. Результат вычисления будет отображаться в окне интегрированной среды newLISP-tk.
10) Выйти из редактора, выбрав в его главном меню File | Exit Editor.

7. Особенности установки newLISP в операционной системе Ubuntu.

Как правило, разработчики программ, ориентированных на Ubuntu, рекомендуют их устанавливать непосредственно из репозиториев. Установка осуществляется с помощью особой программы dpkg – пакетного менеджера для Debian-систем, к которым относится и Ubuntu. Запуск этой программы осуществляется из терминала (Ctrl+Alt+t) командой:

sudo dpkg –i <*имя_deb-пакета_программы*>.deb

Пример (для newLISP-GS версии 10.6.2, 64-битная, 20.01.2015 г.):

sudo dpkg –i newlisp_10.6.2-1_amd64.deb

Для справки: sudo – утилита, которая предоставляет привилегии суперпользователя, называемого root, для выполнения операций администратора системы, в частности, установки программ.

В случае отсутствия в репозитории разработчика требуемого пакета либо его версии, например, под заданную разрядность Ubuntu (на компьютере установлена 32-битная система, а пакет рассчитан на 64-битную), производится установка из deb-пакета, который загружается на жёсткий диск компьютера либо съёмный носитель отдельно из стороннего репозитория. Очевидный минус такого подхода – менеджер обновлений Ubuntu не будет отслеживать появление новых версий установленной из пакета программы. Далее рассмотрим вариант такой установки для newLISP версии 10.3.4-1 (32 бита). В данном примере установка осуществляется из deb-пакета newlisp_10.3.4-1_i386.deb, который предварительно загружается из репозитория по адресу:

http://dpkg.reactor-core.org/ubuntu/dists/natty/universe/binary-i386/newlisp_10.3.4-1_i386.deb

После загрузки deb-пакета из репозитория осуществляется его установка с помощью вышеупомянутой программы dpkg.

sudo dpkg –i <*путь_к_пакету*>/newlisp_10.3.4-1_i386.deb,

где путь к пакету прописывается согласно принятому в Linux синтаксису, например:

/media/shi/Sys_vol2/Install/newlisp,

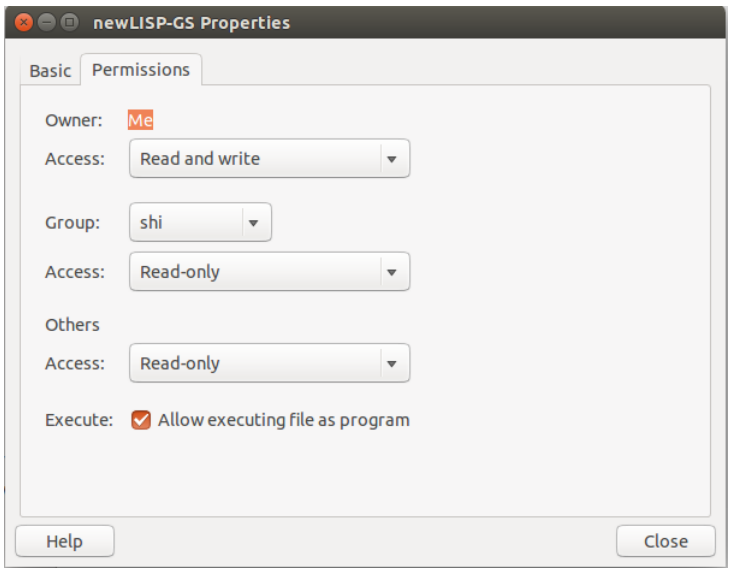
соответственно, newlisp здесь – директория, в которой непосредственно находится пакет, shi – директория с именем пользователя, куда смонтировано запоминающее устройство (в данном случае – том с меткой Sys_vol2, отвечающий одному из разделов жёсткого диска).

Исполняемый файл интерпретатора (newlisp-10.3.4) и символьная ссылка на него будут размещены в директории /usr/bin. Файлы, относящиеся к реализации интегрированной среды, разместятся в созданной при этом директории /usr/share/newlisp-10.3.4.

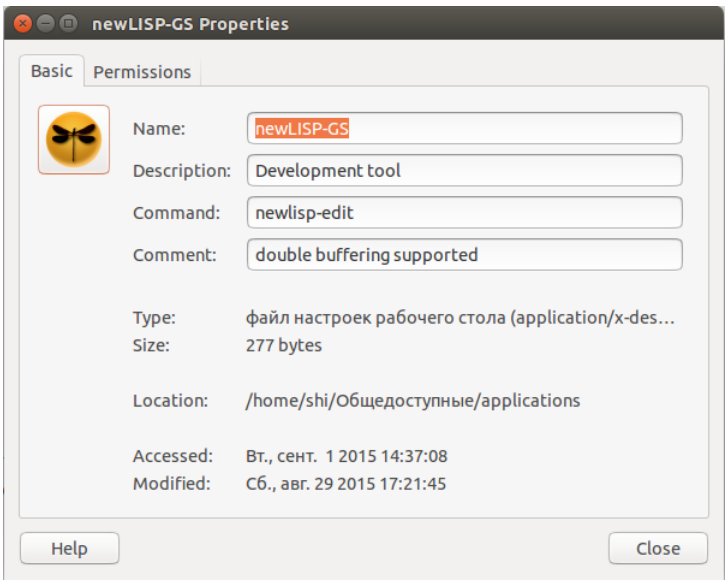
После установки deb-пакета интерпретатор и интегрированная среда newLISP становятся доступными для использования. Интерпретатор можно вызвать из терминала, введя команду newlisp. Работа с интерпретатором в режиме командной строки аналогична реализованной в Windows-версии. Также из терминала можно вызвать и интегрированную среду, введя команду newlisp_edit.

```
[Desktop Entry]
Version=1.0
Name=newLISP-GS
Comment=double buffering supported
GenericName=Development tool
Keywords=newlisp-edit;newlisp;lisp;GS;
Exec=newlisp-edit
Terminal=false
Type=Application
Icon=/usr/share/newlisp-10.3.4/newLISP128.png
Path=
Categories=
NoDisplay=false
```

Рис. 3. Листинг файла newlisp_edit.desktop



а



б

Рис. 4. Окно свойств для newlisp_edit.desktop

Для создания значка запуска интегрированной среды newLISP-GS, размещаемого на панели Unity, нужно создать файл ярлыка, имеющий расширение .desktop, в котором будут указаны все параметры запуска и отображения приложения. Рекомендуемый вариант размещения файла ярлыка – домашняя папка пользователя (т.е. /home/<*имя_пользователя*>),

где в директории с именем Общедоступные (share) создаётся поддиректория applications (если она там не была создана ранее), в которой и размещается указанный файл. Примерный листинг файла ярлыка для newLISP-GS приведён на рис. 3. В данном примере файл был назван newlisp_edit.desktop и помещён в директорию /home/shi/Общедоступные/applications[3].

При написании desktop-файла обязательными являются строки Name (название приложения, используемое для отображения на панели Unity) и Exec (путь к приложению/запуск приложения). Остальные строки допускается оставлять пустыми. Здесь, тем не менее, следует отметить строку Keywords, где задаются те слова, по которым будет осуществляться поиск ярлыка в Главном меню Ubuntu.

Предпоследний шаг: используя файловый менеджер, нужно перейти в директорию

~/ Общедоступные / applications,

где правой кнопкой «мыши» нажать на пиктограмму созданного ярлыка, а в появившемся меню выбрать пункт Properties (Свойства). Далее открывается окно (рис. 4, *а*), в котором во вкладке Permissions необходимо установить флажок рядом с «Allow executing file as program», чтобы пиктограмма изменилась на стандартную для newLISP-GS (рис. 4, *б*).

Созданный значок теперь может быть перемещён «мышью» на боковую панель Unity.

8. Задание на лабораторную работу.

8.1. Ознакомиться с описанием лабораторной работы.
8.2. Выполнить примеры.
8.3. Выполнить свой вариант задания, вариант выдает преподаватель. Задание выполнить различными способами, применяя простейшие и функции из расширения базовых функций Common Lisp (muLISP, newLISP).

Примечание. Помимо Лиспа, допускается выполнение заданий работы на иных языках, реализующих функциональную парадигму, например, Python [8]. При этом свой вариант решения следует в отчёте представлять в сравнении с тем, как бы он был реализован на Лиспе.

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***.

[3] В Linux допускается более короткая запись: ~/Общедоступные/applications

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. Третий подсписок состоит из третьих элементов исходных списков.
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.

Продолжение таблицы 6.

Вариант.	Задача.
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 при 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 списка-первого элемента lst возвращает список из последнего элемента lst в качестве головы и первого элемента lst в качестве хвоста; 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.	Даны лисповские формы expr1 и expr2. Написать функцию, которая сравнивает результаты их вычисления и в случае эквивалентности формирует список из этих форм. В противном случае формируется список из результатов вычисления форм. Вычисление 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, формирующую по двум числам список из трех элементов. Первый – результат целочисленного деления чисел, второй есть результат их умножения, 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.

9. 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;
- выводы по проведенным экспериментам.

Литература.

1. Хювёнен Э. Мир Лиспа. В 2-х т.: пер с финск. / Э. Хювёнен, Й. Сеппянен. М.: Мир, 1990.
2. newLISP™ for Mac OS X, GNU Linux, Unix and Windows [Электронный ресурс]. – URL: http://www.newlisp.org (дата обращения 24.10.2023).
3. GNU CLISP – an ANSI Common Lisp Implementation [Электронный ресурс]. – URL: http://www.clisp.org (дата обращения 24.10.2023).
4. Скриптовый язык newLISP [Электронный ресурс]. – URL: http://www.script-coding.com/newLISP.html (дата обращения 24.10.2023).
5. Русскоязычная документация по Ubuntu: установка программ [Электронный ресурс]. – URL: http://help.ubuntu.ru/wiki/установка_программ (дата обращения 24.10.2023).

6. Новиков П.В. Функциональное программирование на Common Lisp / П.В. Новиков. – М.: Издательство МАИ, 2023. – 96 с [Электронный ресурс]. URL: https://reader.lanbook.com/book/344057#1 (дата обращения 24.10.2023).
7. Петренко А.А. Функциональное программирование / А.А. Петренко, А.О. Суворов, Н.Ф. Шаякбаров. Пермь: ПНИПУ, 2022. – 160 с [Электронный ресурс]. URL: https://reader.lanbook.com/book/328847#1 (дата обращения 24.10.2023).
8. Уроки по языку Python [Электронный ресурс]. URL: https://devpractice.ru/python-lessons/ (дата обращения 24.10.2023).