**Лабораторная работа №1**  **Разработка задач на ПРОЛОГ-е, ЛИСПе**

**Цель работы:**освоить основные возможности и базовые понятия языка ПРОЛОГ  и ЛИСП, изучить основные принципы построения программ на языке функционального и логического программирования.

Любая программа, написанная на Турбо-ПРОЛОГе, состоит из пяти разделов. Ключевые слова: domains, database, predicates, goal, clauses -  отмечают начала соответствующих разделов. Большинство программ не содержит всех пяти названных разделов. Турбо-ПРОЛОГ обеспечивает возможность включения в любом месте программы комментарий любой длины, которые обрамляются символами /\* и \*/ [1,2].

При написании программ важно соблюдать следующие правила: имена всех отношений и объектов записываются со строчной  буквы; сначала записывается имя отношения (предикат). Затем через запятую записываются имена объектов, а весь список имен объектов заключается в круглые скобки - каждый факт, цели должны заканчиваться точкой (.).

Выполнение программы начинается, когда система встретит оператора цели. Цель – это формулировка задачи, которую данная программа должна решить. Турбо-ПРОЛОГ использует как внутренние цели, которые содержатся в программе, так и внешние цели, которые вводятся с клавиатуры после запуска программы. В этом случае Турбо-ПРОЛОГ выдает приглашение Goal (Цель). Турбо-ПРОЛОГ пытается сопоставить цель с фактами и правилами программы. Принцип сопоставления, который необходимо твердо усвоить, – сверху вниз и слева направо. *База данных (БД)) на ПРОЛОГе*. Статические базы данных являются частью кода программы и не могут изменяться во время работы. Динамические базы данных можно дополнять новыми фактами или удалять из них некоторые утверждения. Другая особенность динамической базы состоит в том, что она может храниться в отдельном файле, записана на диск или считана с диска в оперативную память. Для описания  предикатов динамической базы предназначен раздел программы: *database.*Например *dstudent(string, symbol, integer).*

В Турбо-ПРОЛОГе имеется набор встроенных предикатов для изменения фактов динамической базы: *asserta (X)* добавляет факт в начало базы данных; *assertz(X)* добавляет факт в конец базы данных; *retract(X)* удаляет из базы данных факт, сопоставимый с заданным фактом; *consult*(имя файла) открывает файл для добавления фактов в базу данных, размещенную в оперативной памяти [3-5].

*Контрольные примеры на ПРОЛОГе*.Все примеры были взяты из [6].

*Пример 1.**Поиск с возвратом.*Написать программу, реализующую определенную структуру данных. Каждый факт должен содержать не менее трех элементов в кортеже, количество записей – фактов не менее 15. Реализовать вывод всей информации из структуры, поиск по одному и по двум атрибутам записи.

DOMAINS

   territory=ter(real, integer).

   population=pop(real, integer).

   info=c(string, territory, population, string).

PREDICATES

   country(info).

   show.

   search(string).

   search(integer, integer).

CLAUSES

   country(c("Australia", ter(7686.8, 6), pop(21585.1, 52), "Kanberra")).

   country(c("France", ter(674.8, 47), pop(64473.1, 20), "Paris")).

   country(c("India", ter(3287.6, 7), pop(1131191, 2), "New Delhi")).

   country(c("Hungary", ter(93.0, 109), pop(9930.9, 79), "Budapest")).

   country(c("Canadian", ter(9984.7, 2), pop(33091.2, 37), "Ottawa")).

   country(c("China", ter(9570, 3), pop(1322178.2, 1), "Pekin")).

   country(c("Russia", ter(17075.4, 1), pop(141887.5, 9), "Moskow")).

   country(c("USA", ter(9518.9, 4), pop(304000.0, 3), "Washington")).

   country(c("BreatBritain", ter(244.8, 76), pop(60776.2, 21), "London")).

   country(c("Greece", ter(131.9, 94), pop(10964.0, 70), "Athenes")).

   country(c("Kazakhstan", ter(2724.9, 9), pop(15658.3, 61), "Astana")).

   country(c("Madagascar", ter(587.0, 45), pop(19448.8, 55), "Antananarivo")).

   country(c("Maldives", ter(0.3, 186), pop(298.9, 166), "Male")).

   country(c("Estonia", ter(45.0, 129), pop(1342.4, 151), "Tallinn")).

   country(c("Japan", ter(377.8, 60), pop(127433.5, 10), "Tokyo")).

   country(c("Czechia", ter(78.9, 114), pop(10403.1, 79), "Prague")).

     show:-write("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"),

  write("\* COUNTRY\t\*  TERRITORY  \t\*  POPULATION  \t\* CAPITAL \*"),         nl,         write("\*        \t\* km    place \t\* people  place\t\*  \t\*\n"),

         write("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n"), nl,

         country(c(X, ter(At, Bt), pop(Ap, Bp), Y)),

         writef("%s\t\t%-1\t%u\t%-1\t%u\t%s\n", X, At, Bt, Ap, Bp, Y),

               fail.

      search(X):-

         country(c(Y, ter(\_, \_), pop(\_, \_), X)),

         writef("The %s is the capital of the %s.\n", X, Y), fail;

              country(c(X, ter(At, Bt), pop(Ap, Bp), Y)),

         writef("The capital of %s is %s.\nThe territory is %-1, zanimaet %u mesto v mire.\nThe population is %-1, zanimaet %u mesto v mire.\n", X, Y, At, Bt, Ap, Bp), fail.

      search(T, P):-

         country(c(X, ter(\_, Bt), pop(\_, Bp), \_)),

         Bt<=T,  Bp<=P,

         writef(" %s, ", X), fail.

Результаты.Вывод информации на экран. Вывод информации организуется с помощью команды show.

*Поиск по одному атрибуту.*  Поиск можно осуществить по названию страны и по городу. Например, найдем информацию о стране Китай (*China*) и о городе Москва (*Moskow*): *Goal: search ("China").*, и второй запрос *Goal: search ("Moskow ").*

Найдем все страны, которые занимают выше 10-го места по территории и населению и страны, которые занимают выше 100-го места по территории и выше 50-го по населению: *Goal: search (10,10).*, и второй запрос *Goal: search (100,50).*

*Пример 2.**Рекурсия.*Написать рекурсивную программу вычисления суммы ряда чисел *cos(n).* Результат выведите в виде таблицы. Применить нехвостовую и хвостовую рекурсии. Выполнение осуществляется с помощью команды *RUN.*

Выполнение работы.Не хвостовая рекурсия:

PREDICATES

   sum(integer, real).

CLAUSES

   sum(0, 1):-!.

   sum(N, R):-

      Next\_N=N-1,

      sum(Next\_N, P),

      R=cos(N)+P,

      writef(" % \t %-4", N, R), nl.

GOAL

  write("        \*\*\* Lab 3. Recurse \*\*\*"), nl, nl,

  write("Enter quantity of elements of a number: "),

  readint(X), nl,

  write("Number       SumCos"), nl,

  sum(X, Res).

*Хвостовая рекурсия*:

PREDICATES

   sum(integer, real, real).

CLAUSES

   sum(0, \_, \_):-!.

   sum(N, R, P):-

      Next\_N=N-1,

      R=cos(N)+P,

      sum(Next\_N, R, R),

      writef(" % \t %-4", N, R), nl.

GOAL

  write("        \*\*\* Lab 3. Recurse \*\*\*"), nl, nl,

  write("Enter quantity of elements of a number: "),

  readint(X), nl,

  write("Number       SumCos"), nl,

  sum(X, Res, 0).

*Пример 3.* *Обработка списков.* Написать программу для вывода *n*-го элемента списка.

DOMAINS

   list=integer\*.

PREDICATES

   nth\_element(integer, list, integer)

CLAUSES

   nth\_element(1, [A|\_], A):- !.

   nth\_element(N, [\_|L], A):-

      N1=N-1,

      nth\_element(N1, L, A).

GOAL

  write("        \*\*\* Primer 3 \*\*\*"), nl, nl,

  write("Enter number of element of the list: "),  /\*введите номер элемента в списке \*/

  readint(N),

  nth\_element(N, [0, 1, 2, 3, 4], A),

  writef("Element # % = %", N, A), nl.    /\*% обозначает формат вывода \*/

*Результаты.*Выполнение осуществляется с помощью команды *RUN.*

*Пример 4***.***Рекурсивные структуры данных.* Проверить, является ли дерево упорядоченным, уточнить: по возрастанию или по убыванию.

DOMAINS

   tree=t(integer, tree, tree); empty()

PREDICATES

   print\_tree(tree).

   order(tree)

   order\_left(integer, tree)

   order\_right(integer, tree)

CLAUSES

   print\_tree(empty):- !.

   print\_tree(t(R, Left, Right)):-

      write(R, '\t'),

      print\_tree(Left),

      print\_tree(Right).

   order(empty):- !.

   order(t(\_, empty, empty)):- !.

   order(t(R, Left, Right)):-

      order\_left(R, Left),

      order\_right(R, Right),

      order(Left),

      order(Right),

      write("Tree order by vozrast\n");

      order\_left(R, Right),

      order\_right(R, Left),

      order(Left),

      order(Right),

      write("Tree order by ubivan\n").

   order\_left(\_, empty).

   order\_left(T, t(L, \_, \_)):- T>=L.

   order\_right(\_,empty).

   order\_right(T, t(R, \_, \_)):- T<=R.

GOAL

   write("        \*\*\* primer4. Tree \*\*\*"), nl, nl,

   Tree1=t(6, t(3, t(2, empty, empty),

                  t(4, empty, empty)),

             t(7, t(3, empty, empty),

                  t(8, empty, empty))),

   Tree2=t(6, t(7, t(8, empty, empty),

                   t(5, empty, empty)),

              t(3, t(4, empty, empty),

                   t(2, empty, empty))),

   Tree3=t(5, t(2, t(8, empty, empty),

                   t(1, empty, empty)),

              t(4, t(3, empty, empty),

                   t(2, empty, empty))),

   write("Tree: "),

   print\_tree(Tree3), nl,

   order(Tree3),

   write("Tree order!\n"), !;

   write("Tree not order\n").

*Пример 5.* *Базы данных.* Создать базу данных из начальной базы с учетом приведенных условий. Начальная база содержит факты произвольного вида. Удалить все дублирующиеся факты с выводом соответствующих сообщений. Выполнить реализацию внешней базы данных, при этом результат поместить во внешний файл. При реализации внутренней базы данных результат выводится в окно выполнения ПРОЛОГа.

DOMAINS

   list=string\*.

DATABASE

   country(string).

PREDICATES

   import.

   export.

   nondeterm del(string).

   add(list).

CLAUSES

   import:-

      consult("Lab1In.txt"),

      write("   Database import from file!"), nl.

   export:-

      save("Lab1Out.txt"),

      write("   Database export in file!"), nl.

   del(H):- retract(country(H)).

   add([H|T]):-

      country(H), !,

      write("Double fact is delete: ", H), nl,

      add(T).

   add([H|T]):-

      assertz(country(H)), !, add(T).

   add([]).

GOAL

   import,

   findall(H, del(H), T),

   add(T),

   export.

*Результаты*

|  |  |
| --- | --- |
| *Файл Lab1\_Imp.txt:* | *Файл Lab1\_Exp.txt:* |
| country("Russia") | country("Russia") |
| country("USA") | country("USA") |
| country("France") | country("France") |
| country("Breat Britan") | country("Breat Britan") |
| country("Germany") | country("Germany") |
| country("USA") |  |
| country("France") |  |
| country("Germany") |  |

*Пример 6.* *Строки и файлы.* Написать программу, которая бы искала введенное слово в тексте (текст в файле) по максимальному количеству совпадающих первых букв. Результат сохранить в новом файле.

DOMAINS

   Str=string.

   Ch=char.

   Word=string.

   file=f1; f2.

PREDICATES

   find(Word).

   search(string, string, integer).

CLAUSES

   find(Word):-

      openread(f1, "lab7in.txt"),

      openwrite(f2, "lab7out.txt"),

      writedevice(f2),

      file\_str("lab7in.txt", Str),

      str\_len(Word, Len),

      search(Str, Word, Len),

      closefile(f1),

      closefile(f2).

   search(Str, Word, Len):-

      searchstring(Str, Word, Pos),

      writef("The word [%s] find on % position", Word, Pos);

      Len2=Len-1,

      substring(Word,  0, Len2, Word2),

      search(Str, Word2, Len2).

GOAL

   write("      \*\*\* Lab 7. String & File \*\*\*"), nl, nl,

   write("Enter the word: "),

   readln(Word),

   find(Word).

*Программы на ЛИСПе.*ЛИСП ориентирован на обработку нечисловых задач. Он основан на алгебре  списочных структур, лямбда-исчислении и теории рекурсий [7-9]. Язык имеет функциональную направленность, т. е. любое предложение заключенное в скобки, введенное вне редактора считается функцией и выполняется сразу после нажатия «*ENTER*». Чтобы предотвратить вычисление значения выражения, нужно перед этим выражением поставить апостроф «’». Апостроф перед выражением - это сокращение ЛИСПовской функции *QUOTE*. Определения функций могут храниться в файлах и  загружаться  используя  функцию  *LOAD*: *(load <имя файла>).* Эта функция загружает файл выражений и  выполняет  эти выражения. <Имя файла> - это строковая константа, которая представляет собой  имя  файла  без  расширения  (подразумевается  расширение *"\*.lsp"*). Если  операция  успешно завершена, *LOAD* возвращает имя последней функции, определенной в файле. Если операция не выполнена, *LOAD* возвращает имя файла в виде строкового выражения. Функция *LOAD* не может вызываться  из  другой  функции  ЛИСП.  Она должна  вызываться  непосредственно с клавиатуры, в то время как  ни  одна  другая  функция ЛИСП не находится в процессе выполнения. Интерпретатор считает файлами, содержащими исходные тексты программ на ЛИСПе, все файлы, имеющие расширение *LSP*. Запуск системы *XLisp*осуществляется командой: *Xlisp-PLUS*  После нескольких секунд загрузки на экране дисплея появится сообщение: *XLISP-PLUS Version 2.1h Portions Copyright (C )  1988, by David Betz. Modified by Thomas Almy and others*.

После чего появится знак  >, означающий приглашение системы к работе. Системный редактор начинает работать и выдает на экран свое меню: *File, Edit и Run.*

*Функции разбора.* Функция *CAR* возвращает в качестве значения первый элемент списка. (*CAR*  список) ð *S* - выражение (атом либо список). *Функция создания* *CONS*. Функция *CONS* строит новый список из переданных ей в качестве аргументов головы и хвоста. (*CONS* голова хвост). Для того чтобы можно было включить первый элемент функции *CONS* в качестве первого элемента значения второго аргумента этой функции, второй аргумент должен быть списком. Значением функции *CONS* всегда будет список. *Функции присваивания:**SET, SETQ, SETF*. Функция *SET* - присваивает символу или связывает с ним некоторое значение. Причем она вычисляет оба своих аргумента. Установленная связь действительна до конца работы, если этому имени не будет присвоено новое значение функцией *SET*. *Предикаты ATOM, EQ, EQL, EQUAL.* Предикат - функция, которая определяет, обладает ли аргумент определенным свойством и возвращает в качестве значения *NIL* или *T*. Предикат *ATOM* проверяет, является ли аргумент атомом: (*ATOM s* - выражение). Значением вызова *ATOM* будет *T*, если аргументом является атом, и *NIL* - в противном случае. *Список.* Список - упорядоченная последовательность, элементами которой являются атомы либо списки. Списки заключаются в круглые скобки, элементы списка разделяются пробелами. Несколько пробелов между символами эквивалентны одному пробелу. Первый элемент списка называется «головой», а остаток, т. е. список без первого элемента, называется «хвостом». Список,  в котором нет ни одного элемента, называется пустым и обозначается «()» либо *NIL*. Символ - это имя, состоящее из букв, цифр и специальных знаков, которое обозначает какой-нибудь предмет, объект, действие. В языке ЛИСП как для вызова функций, так и для записи выражения принята единообразная префиксная форма записи, при которой как имя функции или действия, так и сами аргументы записываются внутри скобок:  (f x), (g x y), (h x (g y z)) и т. д. Рассмотрим некоторые примеры работы со списками [7,19].

*Пример 1.*Даны два списка. Получить результат произведения этих двух списков в виде списка.  Программа на ЛИСПе имеет следующий вид:

   (terpri)

  (defun  proizvedenie (spisok2   spisok1)

               (cond ((null  spisok1)  0)

               (T  (prince  (\*  (car spisok2)  (car  spisok1))) (prince “ ”)

  (proizvedenie (cdr spisok2)  (cdr  spisok1)))))

  (princ “Введите списки”)

  (terpri)

  (setq a (read))

  (setq b (read))

  (setq c () )

  (terpri)

  (princ “Список в результате  произведения”)

  (terpri)

  (proizvedenie a  b)

  (terpri)

  (princ “Конец !!!”)

*Пример 2.* Даны две цифры. Получить результат сложения  этих двух цифр. Программа на ЛИСПе будет иметь следующий вид:

        (defun sum (N)

    (cond  ((=N 1) 1)

      (T (+ N( sum (-N  1))))))

*Пример 3.* Дан список. Получить результат сложения  элементов списка. Программа на ЛИСПе будет иметь следующий вид:

        (defun sl (L)

           (cond (( Null  L) 0)

              (T( + (car L))

                  (sl(cdr  L))))))

*Пример 4.* Дано число. Получить факториал этого числа. Программа на ЛИСПе будет иметь следующий вид:

         (defun  fact (N)

            (cond ((= N 1) 1)

               (T (\* N(fact(-N 1))))))

*Пример 5.* Даны два списка. Получить результат Чередование двух списков в виде списка.  Программа на ЛИСПе будет иметь следующий вид:

         (defun  sher (x y)

             (cond ((Null x) y)

                (T  (cons (car x)

                     (sher y(cdr x))))))

*Пример 6.*Вычисление суммы N чисел.

(defun sum (N)

    (cond  ((=N 1) 1)

      (T (+ N( sum (-N  1))))))

   (setq N (read))

(terpri)

(princ ' !!!')

*Пример 3.* Выдать на экран среднее число из трех.

(defun sred (x y z)

  (cond ((> x y)

          (cond ((> y z) y)

                    ((> x z) z)

                    (t x)))

          (t(sred y x z ))))

*Пример 7.* Слоны. Задача: На доске 4х4 расставить 4 слона так, чтобы они все находились не под ударом друг друга.

(defun el\_in (x y lst)

 (cond

   ((NULL lst) 0)

   ((and (eql (car(car lst)) x) (eql(car(cdr(car lst))) y))  1 )

   ( 1 (el\_in x y (cdr lst)) ) ))

(defun el\_bit (x y lst)

 (cond

   ((NULL lst) 0)

   ((and (eql (abs (-(car(car lst)) (car(cdr(car lst))))) (abs(- x y)))) 1)

   ( 1 (el\_bit x y (cdr lst)) ) ))

(defun find\_pos1 (lst\_el state)

(cond

 ( (eql (length lst\_el) 4) (print lst\_el))

  ( 1

  (setq nx 0)

  (setq ny 0)

   (loop

     (setq nx (+ nx 1))

     (loop

      (setq ny (+ ny 1))

      (cond

       ((AND (eql (el\_in nx ny lst\_el) 0)  (eql (el\_bit nx ny lst\_el) 0))

             (push nx state)

             (push ny state)

              (find\_pos1 (cons(list nx ny) lst\_el) state)

              (setq ny (pop state))

             (setq nx (pop state)) ) )

      ((> ny 3) (setq ny 0) NULL)  )

     ((> nx 3) NULL)  ) )))

(defun solve()

 (find\_pos1))

**Задания для ПРОЛОГа**

1. Выберите вариант задания и напишите свою программу, на ПРОЛОГе создав статическую базу данных на выбранную тему. Например, напишите программу, использующую предикат: изучает  (фамилия студента, предмет, оценка, срок) -  *study(name,  subject,    numb,  data).*

2.  Напишите утверждения для 6-7 студентов и нескольких предметов. Причем фамилия и предметы, даты могут повторяться. Предикат имеет аргументы символьного и целого числового типов.

3. Напишите программу, содержащую арифметические операции, по варианту ремонта компьютера. Программа должна быть написана для интерактивного режима, т.е. все данные запрашиваются с клавиатуры

4.  Написать программу, которая определяет базу данных как динамическую в отдельном файле.

5.  Написать программу предусматривающей  изменение параметров в базе данных.

6. Разработать на ПРОЛОГе многооконный интерфейс.

**Задания для ЛИСПа**.

1.Проделайте следующие вычисления с помощью ЛИСПа:

а) 3.234∙(45.6+2.43);

б) 55+21.3+1.54∙2.5432-32;

в) (34-21.5676-43)/(342+32∙4.1).

2.Составьте список студентов своей группы  (ФИО ФИО ... ФИО).

3.Для каждого студента а) с помощью функции LIST составьте следующие списки: для самого студента - (дата рождения), (адрес), (средний балл по лекциям), (средний бал по лабораторным). Для отца и матери – (ФИО), (дата рождения), (адрес), (место работы).

4.С помощью CONS и SETQ объедините полученные списки и присвойте их в виде значений символам, означающим ФИО каждого студента: ФИО ст. - (((дата рождения ст.) (адрес ст.)((ср. балл(до десятых) по лекциям).

6.Для каждого студента составьте списки свойств а) оценки по лекциям; б) оценки по практикам; в) оценки по лабораторным работам.

7.Напишите функцию, которая спрашивает у пользователя ФИО студента из группы и выдает следующие данные о нем: год рождения; средний балл; родителей; списки свойств, присвоенные ему раньше.

8.Напишите функцию: от одного аргумента (ФИО любого студента).

**Контрольные вопросы**

1.   Структура программы на Турбо-ПРОЛОГе. Назначение всех разделов.

2.   Связки в предикатах, целевых утверждениях.

3.   Чем отличается повторения от рекурсий в ПРОЛОГе

4.   Какое существует правило рекурсии в ПРОЛОГе.

5.   Перечислите базовые функции ЛИСПа.

6.   Назовите основные отличия предикатов EQ, EQL, EQUAL и =.

7.   Назовите отличия функций CONS и LIST.

8.    Какой элемент называется головой и хвостом списка в ЛИСПе?

9.    Функций определяющие элементы  списка  в ЛИСПе?

10.   Функций которые возвращают  список свойств  в ЛИСПе?

**Интеллектуальные системы**

Одним из важных вопросов, возникающих при проектировании управляющей компоненты систем, основанных на знаниях, является выбор метода поиска решения, т.е. стратегии вывода. От выбранного метода поиска будет зависеть порядок применения и срабатывания правил. Процедура выбора сводится к определению направления поиска и способа его осуществления.