Московский авиационный институт

(государственный технический университет)

**Факультет прикладной математики и физики**

Кафедра вычислительной математики и программирования

**Лабораторная работа №1**

по курсу

«Логическое программирование»

|  |  |
| --- | --- |
| Выполнил: | Карпова В.А. |
| Группа: | 08-306 |
| № по списку: | 9 |
| Руководитель: | Левинская М.А. |
| Оценка: | |
| Дата: | |

Москва

2012 г.

**Условие**

1. Ознакомится с одной из систем программирования на языке Пролог.

2. Проверить наличие в системе программирования встроенных стандартных предикатов обработки списков, отразить их применение в протоколе.

3. Реализовать свои версии стандартных предикатов обработки списков, рассмотренные на занятии (length, member, append, remove, permute, sublist), и убедиться в их работоспособности на ряде различных запросов.

4. Реализовать специальный предикат обработки списка двумя способами: на основе стандартных предикатов обработки списков и без их использования. Отразить в протоколе различные варианты использования предиката на модельных запросах.

5. Реализовать указанный в задании предикат обработки для порядкового представления списка.

6. Реализовать предикат обработки числового списка (списков) для стандартного и порядкового представлений в соответствии с вариантом задания и отразить результат его работы в протоколе.

7. Привести какой-нибудь содержательный пример совместного использования предикатов, реализованных в пунктах 3 и 4.

**Вариант 10**

* Предикаты обработки списков

Получение N-го элемента списка

* Предикаты обработки числовых списков

Лексикографическое сравнение двух списков

**Метод решения**

Для выполнения задания была выбрана реализация SWI-Prolog.

В отличие от GNU Prolog, SWI-Prolog представляет богатый набор возможностей, библиотеки для [многопоточности](http://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C), [юнит-тестирования](http://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C%D0%BD%D0%BE%D0%B5_%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), [GUI](http://ru.wikipedia.org/wiki/GUI), интерфейс к языку программирования [Java](http://ru.wikipedia.org/wiki/Java) и т. д., поддерживает [литературное программирование](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B0%D0%BC%D0%BE%D1%82%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5), содержит реализацию [веб-сервера](http://ru.wikipedia.org/wiki/%D0%92%D0%B5%D0%B1-%D1%81%D0%B5%D1%80%D0%B2%D0%B5%D1%80), средства разработчика (включая [IDE](http://ru.wikipedia.org/wiki/Integrated_Development_Environment) с графическими [отладчиком](http://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BB%D0%B0%D0%B4%D1%87%D0%B8%D0%BA) и [профилировщиком](http://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%84%D0%B8%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0))), и обширную документацию.

***% Реализация собственных версий стандартных предикатов***

**lab1\_0.pl**

length1([],0).

length1([\_|T],N) :- length1(T,N1), N is N1+1.

member1(A,[A|\_]).

member1(A,[\_|T]) :- member1(A,T).

append1([],X,X).

append1([A|T],X,[A|Z]) :- append1(T,X,Z).

remove1(X,[X|T],T).

remove1(X,[A|T],[A|Y]) :- remove1(X,T,Y).

permute1([],[]).

permute1(L,[X|T]) :- remove1(X,L,R),permute1(R,T).

sublist1(L,S) :- append1(\_,L1,L), append1(S,\_,L1).

last1(L,X) :- append1(\_,[X],L).

remove\_n(L,X,N) :- append1(X,Y,L), length1(Y,N).

*% Запросы*

**query1\_0**

write('length: '),

length1([1,q,2,w,r,4,5,t6],X), write(X),ln.

length([],Y),write(Y),ln.

write('member: '),

member1(r,[y,t,e]).

findall(X,member1(X,[u,[y,o],y]),L),write(L).

write('append: '),

append1([[1,2],[i,y],[o,p]],[[111],[90,23,45,67],[appende,liste,printe]],X).

append1(A,[p,y,t,h,o,n],[i,r,o,n,p,y,t,h,o,n]).

write('remove: '),

remove1([9,0],[[4,5],[7,8],[9,0],[a,b]],T).

remove1(a,[b,c,d,r],U).

write('permute: '),

findall(X,permute1([4,5,6],X),L),write(L).

findall(Y,permute1([[[h],[e,l]],[l],o,0],Y),M),write(M).

write('last: '),

last1([9,0,8,0,1],N1),!.

last1([d,f,g,h,j,k,[p,o],[0]],N2),!.

write('remove\_n: '),

remove\_n([],Y,2).

remove\_n([1,2,4,1,6],L,3).

*% Результат*

**proto1\_0**

victoria@fantasy:/media/Study/myMAI/6\_semestr/LP/lab1-2$ swipl -f lab1\_0.pl < query1\_0

% /media/Study/myMAI/6\_semestr/LP/lab1-2/lab1\_0.pl compiled 0.00 sec, 4,284 bytes

Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.8.0)

Copyright (c) 1990-2009 University of Amsterdam.

SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,

and you are welcome to redistribute it under certain conditions.

Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

length:

X = 8.

Y = 0.

member:

false.

[u, [y, o], y]

L = [u, [y, o], y].

append:

X = [[1, 2], [i, y], [o, p], [111], [90, 23, 45, 67], [appende, liste, printe]].

A = [i, r, o, n]

remove:

T = [[4, 5], [7, 8], [a, b]] false.

permute: [[4, 5, 6], [4, 6, 5], [5, 4, 6], [5, 6, 4], [6, 4, 5], [6, 5, 4]]

L = [[4, 5, 6], [4, 6, 5], [5, 4, 6], [5, 6, 4], [6, 4, 5], [6, 5, 4]].

[[[[h], [e, l]], [l], o, 0], [[[h], [e, l]], [l], 0, o], [[[h], [e, l]], o, [l], 0], [[[h], [e, l]], o, 0, [l]], [[[h], [e, l]], 0, [l], o], [[[h], [e, l]], 0, o, [l]], [[l], [[h], [e, l]], o, 0], [[l], [[h], [e, l]], 0, o], [[l], o, [[h], [e, l]], 0], [[l], o, 0, [[h], [e, l]]], [[l], 0, [[h], [e, l]], o], [[l], 0, o, [[h], [e, l]]], [o, [[h], [e, l]], [l], 0], [o, [[h], [e, l]], 0, [l]], [o, [l], [[h], [e, l]], 0], [o, [l], 0, [[h], [e, l]]], [o, 0, [[h], [e, l]], [l]], [o, 0, [l], [[h], [e, l]]], [0, [[h], [e, l]], [l], o], [0, [[h], [e, l]], o, [l]], [0, [l], [[h], [e, l]], o], [0, [l], o, [[h], [e, l]]], [0, o, [[h], [e, l]], [l]], [0, o, [l], [[h], [e, l]]]]

M = [[[[h], [e, l]], [l], o, 0], [[[h], [e, l]], [l], 0, o], [[[h], [e, l]], o, [l], 0], [[[h], [e, l]], o, 0, [l]], [[[h], [e|...]], 0, [l], o], [[[h], [...|...]], 0, o|...], [[l], [...|...]|...], [[...]|...], [...|...]|...].

last:

N1 = 1.

N2 = [0].

remove\_n:

false.

L = [1, 2]

% halt

***% Реализация специального предиката обработки списков.***

% Получение N-го элемента списка

**lab1\_1.pl**

% без использования стандартных предикатов

get1([A|\_],1,A).

get1([\_|T],N,A) :- get1(T,N1,A), N is N1+1.

% на основе использования стандартных предикатов

get2(L,N,R) :- append(X,\_,L),length(X,N),last(X,R).

*% Запросы*

**query1\_1**

get1([n,b,[l,k],0,9,[o]],4,P),!.

get1(['p',r,0,ololo,'G'],5,P),!.

findall(P,get1(['p',r,0,ololo,'G'],N,P),L).

get2(['p',r,0,ololo,'G'],5,P),!.

get2(['p',r,0,ololo,'G'],9,P),!.

get2(['p',r,0,ololo,'G'],0,P),!.

get2(['p',r,0,ololo,'G'],2,P),!.

*% Результат*

**proto1\_1**

victoria@fantasy:/media/Study/myMAI/6\_semestr/LP/lab1-2$ swipl -f lab1\_1.pl < query1\_1

% /media/Study/myMAI/6\_semestr/LP/lab1-2/lab1\_1.pl compiled 0.00 sec, 2,840 bytes

Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.8.0)

Copyright (c) 1990-2009 University of Amsterdam.

SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,

and you are welcome to redistribute it under certain conditions.

Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

P = 0.

P = 'G'.

L = [p, r, 0, ololo, 'G'].

P = 'G'.

false.

false.

P = r.

% halt

***% Лексикографическое сравнение 2х списков***

**lab1\_2.pl**

lower([],[\_|\_]).

lower([X|\_],[Y|\_]) :- X<Y.

lower([A|X],[A|Y]) :- lower(X,Y).

equal([],[]).

equal(A,A).

equal([A|X],[A|Y]) :- equal(X,Y).

bigger([\_|\_],[]).

bigger([X|\_],[Y|\_]) :- X>Y.

bigger([A|X],[A|Y]) :- bigger(X,Y).

cmp(A,B,N) :- lower(A,B), N = 'lower'.

cmp(A,B,N) :- equal(A,B), N = 'equal'.

cmp(A,B,N) :- bigger(A,B), N = 'bigger'.

% Запросы

**query1\_2**

write('compare: '),

cmp([9,0,9,8],[9,0,9],N),!.

cmp([9,0,9],[9,0,9],N),!.

cmp([9,0,9],[],N),!.

cmp([9,0,6,0],[9,0,9],N),!.

cmp([9,0,6,8],[9,0,9],N),!.

% Результат

**proto1\_2**

victoria@fantasy:/media/Study/myMAI/6\_semestr/LP/lab1-2$ swipl -f lab1\_2.pl < query1\_2

% /media/Study/myMAI/6\_semestr/LP/lab1-2/lab1\_2.pl compiled 0.00 sec, 3,528 bytes

Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.8.0)

Copyright (c) 1990-2009 University of Amsterdam.

SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software,

and you are welcome to redistribute it under certain conditions.

Please visit http://www.swi-prolog.org for details.

For help, use ?- help(Topic). or ?- apropos(Word).

compare:

N = bigger.

N = equal.

N = bigger.

N = lower.

N = lower.

% halt

Замечания к lab1\_2.pl

Для реализации лексикографического сравнения 2х списков необходимо уточнить признаки, по которым одно слово считается лексикографически меньше другого слова. Сравнение двух слов происходит последовательно (посимвольно) буква за буквой. И при сравнении каждой буквы смотрим, какой из признаков выполняется. Например, для предиката lower:

1. Если у первого слова достигнут конец раньше, чем у второго, то первое – считается меньшим.
2. Очередная литера в первом слове предшествует в алфавите соответствующей литере во втором слове. Первое слово будет считаться меньшим.
3. Если литеры в обоих словах совпадают, то необходимо сравнивать оставшиеся литеры.

Для bigger все аналогично, с точностью до наоборот.

Для equal очевидно, что выполняются правила:

1. Все литеры должны совпадать
2. Слова имеют одинаковый размер.

**Выводы**

При работе с Прологом мы познакомились с новым синтаксисом, основанном на языке предикатов и логических утверждений. Помимо того, мы узнали о новой парадигме программирования и о новом типе языков высокого уровня: Пролог основан на логической парадигме и является декларативным языком программирования. Логическое программирование основано на теории и аппарате [математической логики](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%BB%D0%BE%D0%B3%D0%B8%D0%BA%D0%B0) с использованием математических методов получения решений. Декларативные языки программирования — это [языки программирования высокого уровня](http://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%81%D0%BE%D0%BA%D0%BE%D1%83%D1%80%D0%BE%D0%B2%D0%BD%D0%B5%D0%B2%D1%8B%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), в которых программистом не задается пошаговый алгоритм решения задачи ("как" решить задачу), а некоторым образом описывается, "что" требуется получить в качестве результата. Основной механизм обработки декларативных утверждений – механизм сопоставления с образцом(Pattern matching), т.е. метод анализа структур данных на наличие в них заданных образцов.

При изучении Пролога поначалу показалось, что Пролог похож на Лисп. И действительно, в функциональном программировании, также как и в логическом, программист остается в неведении о методах, применяемых при вычислении, и последовательности исполнения элементарных действий. Функциональное и логическое программирование являются частями декларативного программирования.