Московский авиационный институт
(государственный технический университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа №1

по курсу
«Логическое программирование»

Выполнил: Баскаков О.А.
Группа: 08-306
№ по списку: 2
Руководитель: Левинская М.А.
Оценка:
Дата:

Москва
2011 г.

Задание

Первоначальное ознакомление с выбранной системой программирования на языке Пролог, реализация предикатов обработки списков в различных представлениях.

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 задания:

Вариант №3.

1. Реверсирование списка
2. Нахождение максимального элемента

Исходные коды на swi-prolog:

```
#! /usr/bin/prolog -s !#

% The length predicate
length1([],0).
length1([_|Tail],Length) :- length1(Tail,TLen), Length is TLen + 1.

% The member predicate
member1(X, [X|_]).
member1(X, [_|Tail]) :- member1(X, Tail).

% The append predicate
append1([], Ys, Ys).
append1([X|Xs], Ys, [X|Zs]) :- append1(Xs, Ys, Zs).

% The remove predicate(var13)
remove1(X, [X|T], T) :-!.
remove1(X, [H|T], [H|S]):- remove1(X, T, S).

% The permutaton predicate
permute1([], []).
permute1([H|T], R):- permute1(T, X), remove1(H,R,X).

% The sublist\2 predicate
sublist1(P, Text) :- append(Zs, _, Text), append(_, P, Zs).

prefix(P,L):-append(P,_,L).
suffix(S,L):-append(_,S,L).
```

```
sublist2(S,L):-prefix(P,L),suffix(S,P).

% The last predicate (var1)
last1(List, Last) :- append(_, [Last], List).

last2([Last|[]], Last).
last2([_|Tail], Last) :- last2(Tail, Last).

% remove last(var2)
last_rm(List2,List) :- append(List, [_], List2).

% The split predicate
split1(List, P, Left, Right) :- append1(Left, [P|Right], List).


% The cycle predicate(var20)
cycle([],[]).
cycle([H|T], List2) :- append(T, [H], List2).

% The head predicate(var8)
head(_,0,[]).
head([],_,[]).
head([H|T], N, [H|T2]):- head(T,N1,T2), N is N1 + 1.

% The reverse predicate(var3)
revappend([], Ys, Ys).
revappend([X|Xs], Ys, Zs) :- revappend(Xs, [X|Ys], Zs).
reverse1(Xs,Ys) :- revappend(Xs,[],Ys).

push1([], A, [A]).
push1([H|T], D, [H|T2]):- push1(T, D, T2).

reverse2([],[]).
reverse2([X|Xs],YsX) :- reverse2(Xs,Ys), push1(Ys, X, YsX).

% reverse order
reversep(X,Z) :- reverse(X,Y), rev1(X,Y,Z).
rev1([],[],[]).
rev1([(X1,_)|Xs],[(_,I2)|Ys],ZX):-rev1(Xs,Ys,Zs),push1(Zs,(X1,I2),ZX).

% The max predicate(var3)
max([H|List], Elem):- max(List, H, Elem).
max([], Elem, Elem).
max([H|T], Z, Elem):- H<Z,!, max(T, Z, Elem).
max([H|T], Z, Elem):- H>=Z, max(T, H, Elem).

% max order
maxp([H|List],I):- mp(List,H,I).
mp([(H,N)|T], (Z,_), Elem):- (H>Z),!, mp(T, (H,N), Elem).
mp([(H,_)|T], (Z,I), Elem):- (H=<Z),!, mp(T, (Z,I), Elem).
mp(_, (_,I), I).
```

Тестирование специальных предикатов:

```
:-op(229,xfx,'==>').

taste_me(A1==>X1, A2==>X2, A3==>X3, A4==>X4) :- write('Testing...'),
     A1 = [1,2,3,4,5,6,7],
     A2 = [(a,1),(c,3),(e,6),(f,7)],
     A3 = [1,2,3,7,5,6,4],
     A4 = [(11,1),(19,3),(13,6)],
%    write('Reverse list'),
     reverse2(A1, X1),
%    write('Reverse order list'),
     reversep(A2, X2),
%    write('Max elem'),
     max(A3, X3),
%    write('Max elem of order list'),
     reverse2(A4, X4).
```

Результаты

```
oleg@debian:~/LP$ ./1.pro
% /home/oleg/LP/1.pro compiled 0.00 sec, 10,848 bytes
Welcome to SWI-Prolog (Multi-threaded, 32 bits, Version 5.10.1)
Copyright (c) 1990-2010 University of Amsterdam, VU 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).

?- taste_me(Rev, Rev_ord, Max, Max_ord).
Testing...
Rev = [1, 2, 3, 4, 5, 6, 7]==>[7, 6, 5, 4, 3, 2, 1],
Rev_ord = [ (a, 1), (c, 3), (e, 6), (f, 7)]==>[ (f, 1), (e, 3), (c,
6), (a, 7)],
Max = [1, 2, 3, 7, 5, 6, 4]==>7,
Max_ord = [ (11, 1), (19, 3), (13, 6)]==>[ (13, 6), (19, 3), (11, 1)].

?- length([1,2,3],X).
X = 3.

?- member(X,[1,2,3]).
X = 1 ;
X = 2 ;
X = 3.

?- append1([1,2],[3,4,5],X).
X = [1, 2, 3, 4, 5].

?- remove1(4,[3,4,5],X).
X = [3, 5] ;
false.

?- permute1([a,s,d],X).
X = [a, s, d] ;
X = [s, a, d] ;
X = [s, d, a] ;
```

```
X = [a, d, s] ;
X = [d, a, s] ;
X = [d, s, a] ;
false.

?- sublist1([a,s],[q,a,s,d]).
true ;
false.

?- sublist2(X,[a,s]).
X = [] ;
X = [a] ;
X = [] ;
X = [a, s] ;
X = [s] ;
X = [] ;
false.

?- last2([12,34,56,7],X).
X = 7 ;
false.

?- last_rm([12,34,56,7],X).
X = [12, 34, 56] ;
false.

?- split1([12,34,56,7],34,X,Y).
X = [12],
Y = [56, 7] ;
false.

?- cycle([5,6,7],X).
X = [6, 7, 5].

?- head([1,2,3,4,5,6,7],4,X).
X = [1, 2, 3, 4] .

?-
% halt
```

Замечания

Можно обратить внимание на тот факт, что если переменная принимает несколько значений, то после всех вариантов выводится false. Этого можно избежать, используя отсечения, но это не всегда дает результат. К тому же false не влияет на другие предикаты. Некоторые предикаты реализуются разными способами в целях оптимизации, в т.ч. получения хвостовой рекурсии и отсечения заведомо ненужных действий.

Выводы

Список позволяет выразить большинства структур данных, т.к. объединяет рекурсивный и итеративный подход к обработке данных. Списки лежат в основе популярного языка LISP, мощь и выразительность которого не вызывают сомнений.
Пролог, будучи декларативным языком программирования, воспринимает в качестве программы некоторое описание задачи или баз знаний и сам производит логический вывод, а также поиск решения задач, пользуясь механизмом поиска с возвратом и унификацией.