Лабораторная работа № 1 по курсу Логическое программирование

Выполнил студент группы М8О-307Б *Лопатин Александр.*

Введение

Связный список — структура данных, состоящая из узлов. Узел содержит данные и ссылку (указатель, связку) на один или два соседних узла. Списки языка Prolog являются односвязными, т.е. каждый узел содержит лишь одну ссылку.

В языке Prolog программист не сталкивается с явной работой с указателями в узлах, однако ему нужно иметь общее представление о списках, т.к. являясь основной структурой данных в функциональных и логических языках, они обладают рядом существенных отличий от массивов, используемых в императивных языках (таких как C++, Java, Pascal). В частности, элемент данных может быть очень быстро добавлен или удален из начала односвязного списка. Однако операция произвольного доступа (обращения к n-ному элементу) в списках выполняется гораздо дольше чем в массивах, т.к. требует n операций перехода по ссылкам.

При работе с односвязными списками необходимо выделять первый узел (называемый головой списка), остальные узлы (составляющие хвост списка) можно получить передвигаясь по указателям вплоть до последнего узла. Хвост списка является таким же списком, как и исходный, поэтому обрабатывается аналогичным образом (рекурсивно).

Реализация стандартных предикатов

Предикат вычисления длины списка:

```
list_length([], 0).
list_length([_|L], N):-list_length(L, M), N is M + 1.
```

Результат работы:

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

Предикат вычисления конкатенации списков:

```
append([], L, L).
append([X|L1], L2, [X|L3]):-append(L1, L2, L3).
```

Результат работы:

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

Предикат принадлежности элемента списку:

```
member(X, [X|_]).
member(X, [_|T]):-member(X, T).
```

Результат работы:

```
?- member(3, [1,2,3,4,5,6]).
true
?- member(-1, [1,2,3,4,5,6]).
false
```

Предикат удаления элемента из списка:

```
remove(X, [X|T], T).
remove(X, [Y|T], [Y|Z]):-remove(X, T, Z).
```

Результат работы:

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

Предикат перестановки элементов в списке:

```
permute([], []).
permute(L, [X|T]):-remove(X, L, Y), permute(Y, T).
```

Результат работы:

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

Предикат подсписков списка:

```
sublist(S, L):-append(_, L1, L), append(S, _, L1).
```

Результат работы:

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

```
X = [1, 2]
X = [1, 2, 3]
X = []
X = [2]
X = [2, 3]
X = []
X = [3]
X = []
```

Предикаты обработки списков

Вариант 3: реверсирование списка.
Без использования стандартных предикатов:

```
reverse([],Z,Z).
 reverse([H|T],Z,Acc) :- reverse(T,Z,[H|Acc]).
```

Результат работы:

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

С использованием стандартных предикатов:

```
reverse_std([],Z,Z).
 reverse_std([H|T],Z,Acc) :-
 append([H], Acc, L1), reverse_std(T, Z, L1).
```

Результат работы:

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

Предикат обработки числовых списков

Вариант 3: нахождение максимального элемента в списке.
Без использования стандартных предикатов:

```
max([X], X).
max([X|Y], X):-
    max(Y, M), X >= M.
max([X|Y], Z):-
    max(Y, Z), X < Z.
```

Результат работы:

```
?- max([1,3,2], X).
X = 3
false
```

Пример совместного использования предикатов, реализованных в предыдущих пунктах

Для примера я выбрал сортировку числового списка - просто на каждом шаге выбираем максимальный элемент, добавляем в новый список, из старого удаляем.

```
sort([], L, L).
sort(L1, C, L2):-
    max(L1, M), remove(M, L1, L3), sort(L3, C, [M|L2]).
```

Результат работы:

```
?- sort([5,2,1,-5,2,7], X, []).
X = [-5, 1, 2, 2, 5, 7]
```

Вывод

В ходе данной лабораторной работы я разобрался, как устроен и как работает язык программирования Prolog, реализовал несложные предикаты для работы со списками.