Московский авиационный институт
(Национальный исследовательский университет)
Факультет прикладной математики и физики
Кафедра вычислительной математики и программирования

Лабораторная работа № 1

Работа со списками и реляционным представлением данных
по курсу "Логическое программирование"

Студент: Алексюнина Ю.В.
Группа: 80-307Б
Преподаватели:
Сошников Д.В., Левинская М.А.
Оценка:

Москва, 2020

Введение

Списки - мощный инструмент для работы с данными в прологе, именно поэтому важно научится с ними работать. По своей структуре и реализации похожи на списки в Лиспе, а если рассматривать императивные языки программирования, то существует сходство со списками в Python и массивами в C и C-подобных языках. Отличие списков в Прологе от списков в императивных языках - это то, как программист получает доступ к данным структуры: в Прологе это делается рекурсивно, а в императивных языках с использованием итераторов. Несмотря на сходства структуры список в Пролог со структурами данных в императивных языках, технология программирования различается до неузнаваемости. В императивных языках программист описывает компьютеру, как получить ответ, а в декларативных он говорит ему то, что ожидает получить. Недаром декларативные и императивные языки - две разных парадигмы.

Реализация стандартных предикатов

А) Вычисление длины списка

```
len([],0).

len([_|T],L):- len(T,L1), L is L1+1.
```

Пример использования:

```
len([1,2,3,5],L).

L = 4
```

Б) Поиск значения в списке

```
mem(X,[X|_]).

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

Пример использования:

```
mem(5,[4,3,5,8]).

true

mem(10,[4,3,5,8]).

false
```

В) Объединение списков

```
app([],X,X).

app([H|T],X,[H|T1]) :- app(T,X,T1).
```

Пример использования:

```
app([a, d], [b, c], L).

L = [a, d, b, c]
```

Г) Удаление элемента из списка

```
rem(X,[X|Tail],Tail).

rem(X,[Head|Tail],[Head|Tail1]) :- rem(X,Tail,Tail1).
```

Пример использования:

```
rem(2,[1,2,3],L).

L = [1, 3]
```

Д) Перестановки

```
perm([],[]).

perm(List,[X|T]) :- rem(X,List,Res), perm(Res,T).
```

Пример использования:

```
perm([4,5,6],L).

L = [4, 5, 6]

L = [4, 6, 5]

L = [5, 4, 6]

L = [5, 6, 4]

L = [6, 4, 5]

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

Е) Вывод всех подсписков списка

```
subl([],[]).

subl(R,L) :- app(_,L1,L),app(R,_,L1).
```

Пример использования:

subl(X,[7,8,9,10]).

X = []

X = [7]

X = [7, 8]

X = [7, 8, 9]

X = [7, 8, 9, 10]

X = []

X = [8]

X = [8, 9]

X = [8, 9, 10]

X = []

X = [9]

X = [9, 10]

X = []

X = [10]

X = []

Предикат обработки списка

2. Удаление последнего элемента списка

Без использования стандартных предикатов:

```
del([P,_], [P]) :- !.
del([H|T], [H|R]) :- del(T, R).
```

Пример использования:

del([1, n, 3, e], L).
L = [1, n, 3]

С использованием стандартных предикатов:

```
del_p(L, X):-reverse(L, [_|T]), reverse(T, X).
```

Пример использования:

```
del_p([1, 3, 5, 7], L).
L = [1, 3, 5]
```

Предикат обработки числовых списков

2. Произведение элементов списка

Без использования стандартных предикатов:

```
mult([],1).
mult([H|T],M):-mult(T,M1),M is H*M1.
```

Пример использования:

```
mult([1,2,3,4],S).
S = 24
```

Пример совместного использования предикатов, реализованных в предыдущих пунктах

Произведение элементов списка, не включая последний.

```
mult_away([],1).
mult_away([H|T],M):- del_p(T, R), mult(R, M1), M is H*M1.
```

Пример использования:

```
mult_away([8, 6, 32, 56], M).
M = 1536
```

Вывод

Эта работа во многом поменяла мое представление о программировании, ведь до этого я никогда не работала с декларативными языками, а только с императивными. Первое знакомство с этим языком было очень непростым, так как с непривычки было очень трудно разобраться, с чем мы имеем дело, однако ближе к завершению работы осознание, медленно приходящее ко мне, позволило закончить начатое.