МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
Одесский национальный университет имени И. И. Мечникова
Институт математики, экономики и механики
Кафедра математического обеспечения компьютерных систем

ОТЧЕТ
о выполнении Лабораторной работы №1:
«Написание программ на языке Prolog»

По предмету: «Экспертные системы»

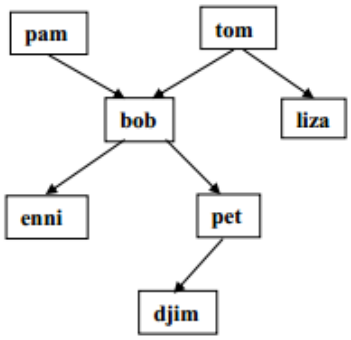
Выполнила студентка
6 курса ФИТ,
Царюк А.О.

Преподаватель
Крапивный Ю. Н.

Одесса – 2017

Задача № 1: Построения программы на языке Prolog.

Предметная область – семейные связи.



Построить запросы:

- Кто мать Боба?
- Кто тетя Пет?
- Кто сестра Боба?
- Кто внук Боба?
- Кто бабушка Джима?

Дополнить программу правилами для определения мамы, тети, бабушки, дедушки.

Код программы:

```
man(bob).
man(tom).
man(djim).
woman(pam).
woman(liza).
woman(pet).
woman(enni).
parents(pam,bob).
parents(tom,bob).
parents(tom,liza).
parents(bob,pet).
parents(pet,djim).
parents(bob,enni).

mother(X,Y):-woman(X),parents(X,Y).
father(X,Y):-man(X),parents(X,Y).
sister(X,Y):-woman(X),parents(Z,Y),parents(Z,X).
aunt(X,Y):-woman(X),parents(Z,Y),sister(X,Z).
son(X,Y):-man(X),parents(Y,X).
daughter(X,Y):-woman(X),parents(Y,X).
grandma(X,Y):-mother(X,Z),parents(Z,Y).
grandpa(X,Y):-father(X,Z),parents(Z,Y).
grandson(X,Y):-son(X,Z),parents(Y,Z).
```

Результат работы программы:

```
?- parents(bob,X).
X = pet ;
X = enni.

?- parents(X,bob).
X = pam ;
X = tom.
```

Задача № 2: Рекурсивный метод программирования.

Необходимо добавить к задаче о родственных связях еще одно отношение – предок.

Код программы:

```
ancestor(X,Y):-parents(X,Y).
ancestor(X,Y):-parents(X,Z),ancestor(Z,Y).
```

Результат работы программы:

```
?- ancestor(tom,pet).
true .

?- ancestor(bob,djim)
true .
```

Задача № 3: Списки данных.

Написать программы, реализующие следующие операции для списков:

a) Вернуть последний элемент списка;
b) Добавить элемент в список;
c) Удалить элемент из списка;
d) Перевернуть список.

Код программы:

```
last(X,Y):-append(_,[X],Y). %вернуть последний элемент списка;

add(X,Y,Z):-append([X],Y,Z).%добавить элемент в начало списка;

remove(X,Y,Y):-not(member(X,Y)),!.%удалить элемент из списка;
remove(X,Y,Z):-append(L1,[X|L2],Y),append(L1,L2,Z),!.

revers([],[]). %перевернуть список (revers);
revers([H|T],Y):-revers(T,Z),append(Z,[H],Y).
```

Результат работы программы:

```
?- last(X,[3,5,6,7,8]).
X = 8 .

  ?- add(1,[3,5,6],X).
  X = [1, 3, 5, 6].

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

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

Задача № 4: Вычислительные возможности языка ПРОЛОГ

Написать программы, реализующие операции для списков:

a) Нахождение суммы элементов в числовом списке;
b) Определение четности длины произвольного списка;
c) Нахождение максимального элемента в числовом списке;
d) Нахождение количества отрицательных элементов в числовом списке;
e) Нахождение в произвольном списке количества атомов совпадающих с атомом X.

Код программы:

```
sum1([],S,S):-!.%нахождение суммы элементов в числовом списке
sum1([H|T],C,S):-C1 is C+H,sum1(T,C1,S).
sum(L,S):-sum1(L,0,S).

oddity([],0):-!.  %определение четности длины произвольного
списка,
oddity([_|[]],1):-!.
oddity([_|[_|T]],R):-oddity(T,R).

maximum([],0):-!. %нахождение максимального элемента в списке
maximum([H|T],X):-maximum1(T,H,X).
maximum1([],C,C):-!.
maximum1([H|T],C,X):-H<C,!,maximum1(T,C,X).
maximum1([H|T],_,X):-maximum1(T,H,X).

negative([],0):-!.%нахождение количества отрицательных элементов
в числовом списке
negative([X|T],K):-X<0,!,negative(T,K1),K is K1+1.
negative([_|T],K):-negative(T,K).

same([],_,0):-!.  %нахождение в произвольном списке количества
атомов совпадающих с атомом X
same(T,X,Y):-same1(T,X,0,Y).
same1([],_,C,C):-!.
same1([X|T],X,C,Y):-!,C1 is C+1,same1(T,X,C1,Y).
same1([_|T],X,C,Y):-same1(T,X,C,Y).
```

Результат работы программы:

```
?- sum([1,2,3,4,5],X).
X = 15.

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

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

?- negative([1,2,3,4,5,-6,-8,-2,-5],X).
X = 4.

 ?- same([1,2,3,4,5,-6,-8,-2,5],5,X).
 X = 2.
```

Задача № 5: Построение процедур обработки списков

a) Утроить каждый компонент списка
b) Все «a», стоящие на четных местах заменить на “j”
c) В заданном списке удалить каждый второй элемент
d) В заданном списке поменять местами каждый элемент, стоящий на четном месте, с предыдущим
e) Заданы два списка одинаковой длины. Сцепить их так, чтобы элементы обоих списков чередовались
f) В заданном символьном списке удалить все элементы ‘a’
g) Все вхождения идущих подряд одинаковых элементов списка заменить одним

Код программы:

```
% 5/1 Утроить каждый компонент списка.
% получитьсписок [a, a, a, b, b, b, c, c, c]
triple([],[]):-!.
triple([H|T],R):-triple(T,R1),append([H,H ,H],R1,R).

%5/2 Все 'a', стоящие на четных местах, заменить на 'j
 %replace(A,B,List,Res)
replace(A,B,L,R):-replace(A,B,L,1,R).
replace(_,_,[],_,[]):-!.
replace(A,B,[H|T],1,[H|R]):-replace(A,B,T,0,R),!.
replace(A,B,[A|T],0,[B|R]):-!,replace(A,B,T,1,R).
replace(A,B,[H|T],0,[H|R]):-!,replace(A,B,T,1,R).

%5/3 В заданном списке удалить каждый второй элемент
rem_even([],[]):-!.
rem_even([H|[]],[H]):-!.
rem_even([H|[_|T]],[H|R]):-rem_even(T,R).

% 5/4 В заданном списке поменять местами каждый элемент, стоящий
на четном месте, с предыдущим
switch_pairs([],[]):-!.
switch_pairs([H|[]],[H]):-!.
```

```
switch_pairs([H1|[H2|T]],[H2|[H1|R]]):-switch_pairs(T,R).

% 5/5 Заданы два списка одинаковой длины. Сцепить их так, чтобы элементы обоих списков чередовались
twolist([],[],[]):-!.
twolist([H1|T1],[H2|T2],R):-
twolist(T1,T2,R1),append([H1,H2],R1,R).
%5/6 В заданном символьном списке удалить все элементы ‘a’
%dell_all(X,L,L1).
dell_all(_,[],[]).
dell_all(X,[X|T],L):-!,dell_all(X,T,L).
dell_all(X,[Y|T],[Y|L]):-!,dell_all(X,T,L).

% 5/7 Все вхождения идущих подряд одинаковых элементов списка заменить одним
rem_repeat([],[]):-!.
rem_repeat([H|T],R):-rem_rep(T,H,R).
rem_rep([],H,[H]):-!.
rem_rep([H|T],H,R):-!,rem_rep(T,H,R).
rem_rep([H|T],X,[X|R]):-!,rem_rep(T,H,R).
```

Результат работы программы:

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

?- replace(5,0,[1,5,3,5,5,-6,-8,5,-2,5],X).
X = [1, 0, 3, 0, 5, -6, -8, 0, -2|...].

?- rem_even([-3,4,-5,-41,4,5],X).
X = [-3, -5, 4].

?- switch_pairs([-3,4,-5,-41,4,5],X).
X = [4, -3, -41, -5, 5, 4].

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

?- dell_all(5,[1,5,3,5,5,-6,-8,5,-2,5],X).
X = [1, 3, -6, -8, -2] .

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

Выводы: в ходе выполнения лабораторной работы были изучены принципы построения программ на языке ПРОЛОГ и рассмотрены такие понятия как: рекурсия, вычислительные операции, списки и процедуры обработки списков.