Задание 1

На базе имеющихся предикатов, описывающих родственные отношения: ***man(symbol), woman(symbol), parent(symbol,symbol), Kol\_det(symbol, symbol, integer), mother(symbol), father(symbol)***

Определить предикаты ***брат, сестра, дядя, бабушка***.

**Исходный код:**

man(mikhail).

man(vitaly).

man(lesha).

woman(marina).

woman(elena).

woman(luydmila).

woman(galina).

parent(marina, mikhail).

parent(vitaly, mikhail).

parent(luydmila, marina).

parent(luydmila, elena).

parent(luydmila, lesha).

parent(galina, vitaly).

% Is X C's sister (and so on)

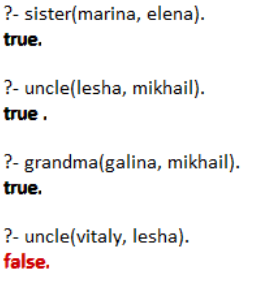
sister(X, C) :- parent(P, X), parent(P, C), woman(X), X \= C.

brother(X, C) :- parent(P, X), parent(P, C), man(X), X \= C.

grandma(X, C) :- parent(P, C), parent(X, P), woman(X).

uncle(X, C) :- parent(P, C), brother(X, P). % uncle is the brother of one's parent

**Результаты:**



Задание 2

Имеется информационная база о ***служащих*** предприятия в виде:

идентификационный номер служащего, фамилия, должность, зарплата, номер отдела. Дополнительно отдельно хранятся данные о ***семейном положении*** служащего в виде: идентификационный номер служащего, фамилия жены или мужа, число детей.

Написать программу, содержащую пример такой базы и дополнить её правилами:

1. ***А*** ***знаком*** с ***В*** , если они работают в одном отделе
2. Жена/муж ***А*** ***знаком*** с женой/мужем ***В*** , если ***А*** и ***В*** работают в одном отделе.
3. ***Составить целевые запросы к программе***:

- Выдать всех служащих первого отдела)

- Определить работников, получающих зарплату более заданной

- Определить работников, имеющих более одного ребёнка.

**Исходный код:**

employee(0, pupkin, slesar, 16000, 666).

employee(1, svechkina, boss, 35000, 1).

employee(2, shkiryak, dvornik, 9000, 1).

employee(3, bubenov, sysadmin, 25000, 2).

employee(4, zyamyatina, buhgalter, 40000, 3).

employee(5, arzamaschev, analitic, 30000, 4).

employee(6, kashtanova, consultant, 15000, 3).

family(0, svechkina, 1).

family(1, pupkin, 1).

family(2, zyamyatina, 0).

family(4, shkiryak, 0).

family(5, kashtanova, 2).

family(6, arzamaschev, 2).

aFamiliarToB(A, B) :- employee(\_, A, \_, \_, DEPN1),

employee(\_, B, \_, \_, DEPN2),

DEPN1 = DEPN2,

A \= B.

spousesFamiliar(A\_SP, B\_SP) :- employee(ID1, A, \_, \_, D),

employee(ID2, B, \_, \_, D),

family(ID1, A\_SP, \_),

family(ID2, B\_SP, \_),

A \= B.

firstDepartment(ID, F) :- employee(ID, F, \_, \_, 1).

salaryMoreThan(F, SALARY\_MIN) :- employee(\_, F, \_, S, \_),

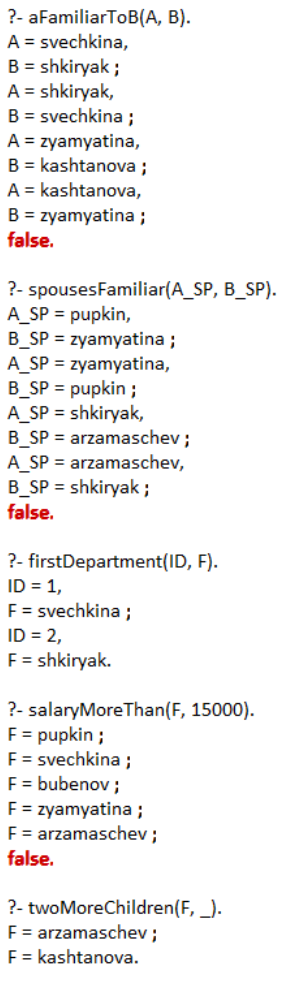
S > SALARY\_MIN.

twoMoreChildren(F, N) :- employee(ID, F, \_, \_, \_),

family(ID, \_, N),

N > 1.

**Результаты:**



Задание 3

Составить программу, вычисляющую значение функции

***f(x,y)=3\*y \* a(x) + sin(x\*y)\* b(y),*** где

***a(x)=sqrt(| x|),***

***b(y)=cos(y) +2.***

Вывести промежуточные значения ***a(x), b(y)***  и результат ***f (x,y)***. Использовать два способа – вычисление в одном предикате и выделение нахождения ***a(x)*** и  ***b(y),***  в отдельные предикаты.

**Исходный код:**

a(X, R1) :- R1 is sqrt(abs(X)).

b(Y, R2) :- R2 is cos(Y) + 2.

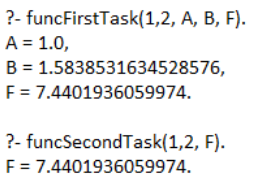
funcFirstTask(X, Y, A, B, F) :- a(X, A),

b(Y, B),

F is 3\*Y \* A + sin(X\*Y) \* B.

funcSecondTask(X, Y, F) :- F is 3\*Y \* sqrt(abs(X)) + sin(X\*Y) \* (cos(Y) + 2).

**Результаты:**



Задание 4

Написать программу, вычисляющую значение функции вида:

|  |  |  |  |
| --- | --- | --- | --- |
| ***f (x,y)*** | ***=*** |  | ***2\*x, если x+y < -1***  ***cos(x\*y), если -1 ≤ x+y ≤ 1***  ***sqrt(x+y), если x+y > 1*** |

**Исходный код:**

func(X, Y, R) :- X + Y < -1,

R is 2\*X.

func(X, Y, R) :- X + Y > 1,

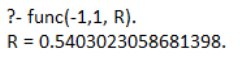
R is sqrt(X + Y).

func(X, Y, R) :- X + Y =< 1,

X + Y >= -1,

R is cos(X \* Y).

**Результаты:**



Задание 5

Написать программу, осуществляющую сравнение двух чисел ***А*** и ***В*** и формирующую одно из сообщений: «***А больше В», «А меньше В», «А равно В»***

**Исходный код:**

func(X, Y, R) :- X + Y < -1,

R is 2\*X.

func(X, Y, R) :- X + Y > 1,

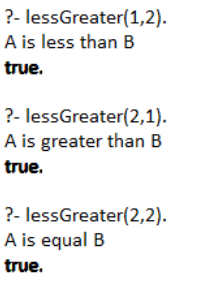
R is sqrt(X + Y).

func(X, Y, R) :- X + Y =< 1,

X + Y >= -1,

R is cos(X \* Y).

**Результаты:**



Задание 6

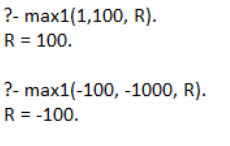
Составить предикат ***max1(x,y,z)***, определяющий ***max*** из двух чисел ***x*** и ***y***

**Исходный код:**

max1(X, Y, Z) :- X > Y -> Z is X ;

Z is Y .

**Результаты:**



Задание 7

Составить предикат ***max1(x,y,z)***, определяющий ***max*** из двух чисел ***x*** и ***y***

**Исходный код:**

disc(A, B, C, D) :- D is B ^2 - 4 \* A \* C.

findRoots(A, B, C, D, X1, X2) :- disc(A, B, C, D),

(D < 0 -> write("there is no root") ;

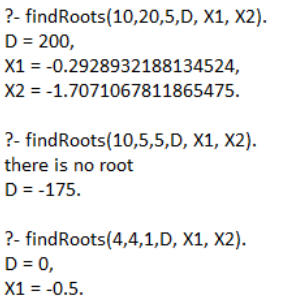
D == 0 -> X1 is (0 - B) / (2\*A) ;

X1 is (0 - B + sqrt(D)) / (2 \* A),

X2 is (0 - B - sqrt(D)) / (2 \* A)

).

**Результаты:**



Задание 8

Имеются сведения о возрасте различных людей в форме:

***возраст(фамилия, число\_лет)*** и их поле: ***мужчина(фамилия)***, ***женщина(фамилия)***.

Используя ***внутрипрограммную*** цель получить ***список всех мужчин старше 18 лет.***

**Исходный код:**

age(vasya, 23).

age(sonya, 20).

age(petya, 30).

age(elena, 16).

age(egor, 12).

man(vasya).

man(petya).

man(egor).

woman(sonya).

woman(elena).

:- initialization((findMenElder18(A, N))).

findMenElder18(Age, Name) :-

age(Name, Age),

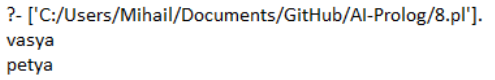
man(Name),

Age > 18,

writeln(Name),

fail.

**Результаты:**



Задание 9

Имеются данные о том, кто кого победил во время соревнований в виде: ***победил(фамилия, фамилия).***

Используя предикат ***победил*** и предикат ***cut,*** необходимо классифицировать соревновавшихся следующим образом:

***если***   ***Х*** победил кого-либо ***и*** *Х* был кем-то побеждён, ***то*** ***Х – боец***

***иначе если*** ***Х***  победил кого-либо, ***то*** ***Х* – *победитель***

***иначе если*** ***Х*** был кем-то побеждён*,* ***то*** ***Х – спортсмен***

**Исходный код:**

% fightes: lozanov, gagarin, kalchenko

% winners: yakov, dubrovskiy, yanvarev, romazanov

% sportsmen: chiolkovskiy, algorin, kodenko

win(lozanov, gagarin).

win(yakov, lozanov).

win(gagarin, algorin).

win(dubrovskiy, kodenko).

win(romazanov, kalchenko).

win(yanvarev, chiolkovskiy).

win(kalchenko, gagarin).

classifyAthletes() :- win(X, \_), win(\_, X),

write(X), writeln(" - fighter").

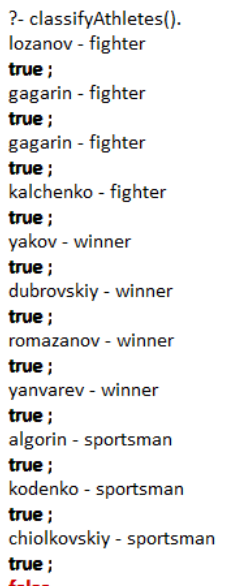
classifyAthletes() :- win(X, \_), not(win(\_, X)),

write(X), writeln(" - winner").

classifyAthletes() :- win(\_, X), not(win(X, \_)),

write(X), writeln(" - sportsman").

**Результаты:**



Задание 10

Пусть имеется информация о возрасте некоторых людей в виде:

***возраст(имя, число лет)*** и их поле: ***мужчина(имя)***, ***женщина(имя)***, цвете волос: ***цвет\_волос(имя, цвет)*** а также ***о различных городах***, когда-либо посещавшихся данными людьми. Необходимо составить программу, осуществляющую поиск и выбор ***одного рыжеволосого мужчины старше 18 лет*** из списка и ***вывод всех городов, которые он посещал***. Целевой запрос должен быть ***внутрипрограммным***.

**Исходный код:**

man(vasya).

man(petya).

man(sergey).

man(andrey).

woman(sveta).

woman(nastya).

age(vasya, 20).

age(petya, 18).

age(sergey, 31).

age(andrey, 23).

age(sveta, 22).

age(nastya, 35).

hairColor(vasya, blond).

hairColor(petya, gray).

hairColor(sergey, redhaired).

hairColor(andrey, redhaired).

hairColor(sveta, blond).

hairColor(nastya, brunette).

cityVisited([ufa], vasya).

cityVisited([samara], petya).

cityVisited([kransodar, moscow], sergey).

cityVisited([st-petersburg], andrey).

cityVisited([habarovsk], sveta).

cityVisited([kazan], nastya).

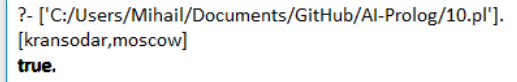
findRedHairedMan() :- man(N), age(N, A), A > 18,

hairColor(N, redhaired),

cityVisited(C, N), !, writeln(C).

:- initialization((findRedHairedMan())).

**Результаты:**



Задание 11

Составить программу, определяющую ***х***  в степени  ***n***, где ***х, n*** - целые числа, ***n≥0.***

**Исходный код:**

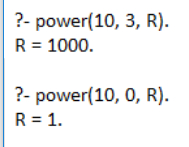
power(\_, 0, 1) :- !.

power(X, N, F) :- N1 is N-1,

power(X, N1, FTEMP),

F is FTEMP \* X .

**Результаты:**



Задание 12

Последовательность Фибоначчи имеет вид: ***0,1,1,2,3,5,8….*** и определяется формулами: ***f(1) = 0, f(2) = 1,…., f(k )= f(k-2) + f(k-1).***

Составить программу, определяющую ***число Фибоначчи*** с заданным номером.

**Исходный код:**

fib1(N, F) :- fib2(N, F, 1, 1, 0).

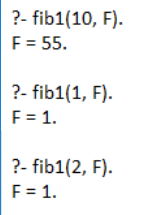
fib2(N, F, N, F, \_) :- !.

fib2(N, F, K, F1, F0) :- K1 is K+1,

F2 is F0+F1,

fib2(N, F, K1, F2, F1).

**Результаты:**



Задание 13

Имеются факты, описывающие родственные отношения между людьми, указывая, кто является чьим родителем: ***родитель(X,Y)***, т.е. ***X*** – родитель ***Y***.

Описать предикат, позволяющий определить, кто является чьим потомком:

***потомок(А,В),*** т.е. ***А*** - потомок ***В***.

**Исходный код:**

parent(marina, mikhail).

parent(vitaly, mikhail).

parent(sergey, marina).

parent(luydmila, marina).

parent(galina, vitaly).

parent(mikhail\_g, vitaly).

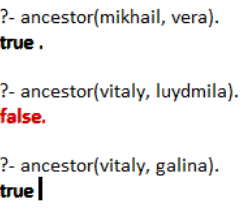
parent(vasya, galina).

parent(vera, galina).

ancestor(A, B) :- parent(B, A).

ancestor(A, B) :- parent(Z, A), ancestor(Z, B).

**Результаты:**



Задание 14

Определить предикат ***lensp(N,L)*** для нахождения длины списка ***L***.

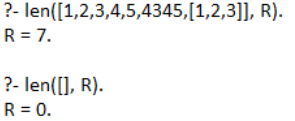
**Исходный код:**

len([], 0).

len([\_|TAIL], LENGTH) :- len(TAIL, LTEMP),

LENGTH is LTEMP + 1.

**Результаты:**



Задание 15

Составить программу для поиска ***суммы всех четных элементов*** списка

**Исходный код:**

getEvenNumbersSum(N) :- sumEvenNumbers(N, Sum), !,

writeln(Sum).

sumEvenNumbers([], Sum) :- Sum is 0.

sumEvenNumbers([H|T], Sum) :- sumEvenNumbers(T, SumTemp),

isOdd(H, R),

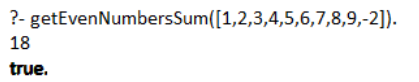
(R == 0, Sum is SumTemp + H;

Sum is SumTemp

).

isOdd(X, R) :- R is X mod 2.

**Результаты:**



Задание 16

Составить программу для перевода списка, записанного в виде цифр, в список из соответствующих слов, например:

**[*3,5,1,3*]** преобразовать в **[*“три”, “пять”, “один”, “три”*]**

**Исходный код:**

convert(0, "zero").

convert(1, "one").

convert(2, "two").

convert(3, "three").

convert(4, "four").

convert(5, "five").

convert(6, "six").

convert(7, "seven").

convert(8, "eight").

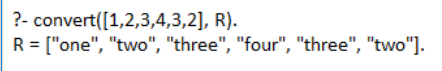
convert(9, "nine").

convert([],[]).

convert([H|T], [X|Y]) :- convert(H, X),

convert(T, Y).

**Результаты:**



Задание 17

Написать программу для определения ***максимального значения*** в списке.

**Исходный код:**

runFindMax([H|T]) :- findMax(T, H).

findMax([], Max) :- write(Max).

findMax([H|T], Max) :- Max > H, findMax(T, Max) ;

findMax(T, H).

**Результаты:**



Задание 18a

Составить программу для определения, является ли первый список ***подсписком*** второго.

**Исходный код:**

belong(Element, [Element|\_]) :- !.

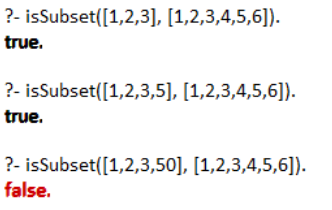
belong(Element, [\_|T]) :- belong(Element, T).

isSubset([], \_).

isSubset([Element|T], List2) :- belong(Element, List2),

subset(T, List2).

**Результаты:**



Задание 18b

Второй вариант предыдущей задачи: определить подсписок через предикат объединения списков ***append***.

**Исходный код:**

isSubset([H|T], List2) :- append(L, [H|\_], List2), !,

last(T, Tast), !,

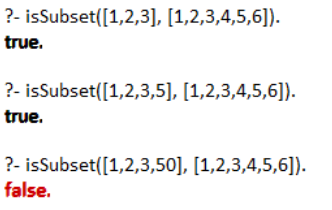
append(\_, [TLast|R], List2), !,

append(L, [H], LH),

append(LH, T, M),

append(M, R, List2).

**Результаты:**



Задание 19

Написать программу, позволяющую найти список, являющийся ***пересечением*** двух списков.

**Исходный код:**

list([], \_) :- !.

list([H|T], L) :- repea(H, L), list(T, L).

repea(\_, []) :- !.

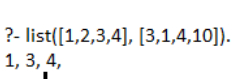
repea(K, [H|T]) :- K \= H, repea(K, T).

repea(K, [H|\_]) :- K is H,

write(K),

write(", ").

**Результаты:**



Задание 20

Написать программу, позволяющую определить ***длину строки***, введённой пользователем, не используя стандартного предиката ***str\_len***.

**Исходный код:**

calcStringLen(Str, Len) :- atom\_chars(Str, Vars),

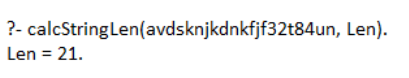
stringLen(Vars, Len).

stringLen([], 0).

stringLen([\_|T], Len) :- stringLen(T, LenTemp),

Len is LenTemp + 1.

**Результаты:**



Задание 21

Подсчитать, сколько ***символов*** находится ***в файле***, определенном пользователем

**Исходный код:**

main :- open("C:/Users/Mihail/Documents/GitHub/AI-Prolog/lab21.txt", read, Str),

readFile(Str, Lines),

calcFileLength(Lines, Len),

close(Str),

writeln(Len).

calcFileLength([], 0).

calcFileLength([H|T], Len) :- calcFileLength(T, LenTemp),

calcStringLen(H, StringLenTemp),

Len is LenTemp + StringLenTemp.

calcStringLen(Str, Len) :- atom\_chars(Str, Vars),

stringLen(Vars, Len).

stringLen([], 0).

stringLen([\_|T], Len) :- stringLen(T, LenTemp),

Len is LenTemp + 1.

readFile(Stream, []) :- at\_end\_of\_stream(Stream).

readFile(Stream, [X|L]) :- \+ at\_end\_of\_stream(Stream),

read\_string(Stream, "\n", "\r\t", \_, X),

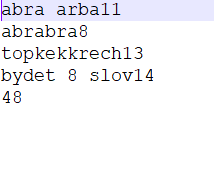
readFile(Stream, L).

% https://stackoverflow.com/questions/4805601/read-a-file-line-by-line-in-prolog

**Результаты:**



**Файл:**



Задание 22

Подсчитать, сколько ***слов*** находится ***в файле***, определенном пользователем

**Исходный код:**

main :- open("C:/Users/Mihail/Documents/GitHub/AI-Prolog/lab21.txt", read, Str),

readFile(Str, Lines),

fileWordsCounter(Lines, N),

close(Str),

writeln(N).

fileWordsCounter([], 0).

fileWordsCounter([H|T], N) :- fileWordsCounter(T, NTemp),

stringWords(H, StringNTemp),

N is NTemp + StringNTemp.

stringWords([], 0).

stringWords([\_|T], N) :- stringWords(T, NTemp),

N is NTemp + 1.

readFile(Stream, []) :- at\_end\_of\_stream(Stream).

readFile(Stream,[X|L]) :- read\_string(Stream, "\n", "\r\t", \_, Y),

split\_string(Y, " ", " ", X),

readFile(Stream, L).

**Результаты:**

(Файл из предыдущей ЛР)

Задание 23

Файл ***tree.dat*** содержит базу данных о плодовых деревьях: яблонях и грушах, включая сорт и тип плодоношения:

***яблоня(”Папировка”,”летний”)***

***яблоня(”Мельба”,”летний”)***

***яблоня(”Антоновка”,”зимний”)***

***яблоня(”Уэлси”,”зимний”)***

***груша(”Лада”,”летний”)***

***груша(”Московская”,”летний”)***

***груша(”Чижовскаяа”,”осенний”)***

Необходимо запросить у пользователя дерево и тип плодоношения и выдать ему все подходящие сорта. В случае отсутствия подходящего сорта выдать ответ ***«Нет данных»***

**Исходный код:**

main :- consult("C:/Users/Mihail/Documents/GitHub/AI-Prolog/tree.dat"),

write("Tree name: "),

read(Tree),

writeln("Type name: "),

read(Type),

%checkSortes(Tree, Type),

runFindSortes(Tree, Type).

runFindSortes(apple, Type) :- appleTree(X, Type),

writeln(X),

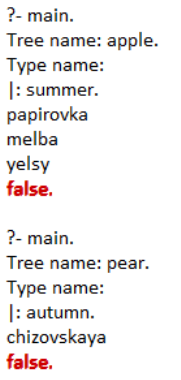
fail.

runFindSortes(pear, Type) :- pearTree(X, Type),

writeln(X),

fail.

**Результаты:**



Задание 24

В файле ***tel.dat*** разместить данные об именах и номерах телефонов, используя предикат ***телефон(имя, номер):***

***телефон(”Иван”,”1234337”)***

***телефон(”Петр”,”1111567”)***

***телефон(”Маша”,”3222232”)***

Составить программу, позволяющую проводить дополнение базы данных, с помощью примерного диалога:

***Введите имя: Толя***

***Введите телефон: 1234567***

***Продолжить? да/нет***

По окончании дополнения базы следует сохранить обновлённый файл ***tel.dat.***

**Исходный код:**

addPhone() :- consult("C:/Users/Mihail/Documents/GitHub/AI-Prolog/phones.dat"),

write("enter the name: "),

read(Name),

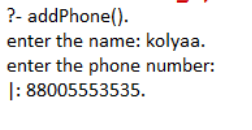
writeln("enter the phone number: "),

read(Number),

assertz(phone(Name, Number)),

open("C:/Users/Mihail/Documents/GitHub/AI-Prolog/phones.dat", write, S), set\_output(S), listing(phone), close(S).

**Результаты:**



**Файл:**

