## **Исходный код**

% Studentexerciseprofile

:-set\_prolog\_flag(occurs\_check, error). % disallow cyclic terms

:-set\_prolog\_stack(global, limit(8 000 000)). % limit term space (8Mb)

:-set\_prolog\_stack(local, limit(2 000 000)). % limit environment space

% Your program goes here

singer("Martishka",instruments(["Skripka"]),"неумеет").

singer("Osel",instruments(["Alt"]),"неумеет").

singer("Kozel",instruments(["Skripka"]),"неумеет").

singer("Mishka",instruments(["Bass"]),"неумеет").

singer("Solovey",instruments([]),"умеет").

no\_musicians(Q,Z):- singer(Q,\_,Z),Z="неумеет".

member(Elem, [Elem|\_Tail]).

member(Elem, [\_Head|Tail]):-

member(Elem, Tail).

instrument(W,R):- singer(W,instruments(E),\_),member(R,E).

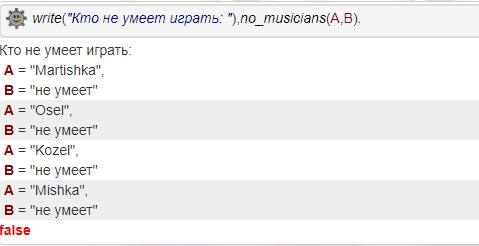
/\*\* <examples> Your example queries go here, e.g.

?- write("Ктонеумеетиграть: "),no\_musicians(A,B).

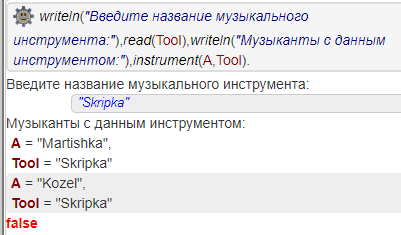
?-writeln("Введитеназваниемузыкальногоинструмента:"),read(Tool),writeln("Музыкантысданныминструментом:"),instrument(A,Tool).

\*/

Вывод 1-ого запроса



Вывод 2-го запроса



# **Лабораторная работа №2**

Функциональное и логическое программирование

**Задание 1.1**

Выполнить программы 1 и 2

ПРОГРАММА 1

song:- write('бегу, '),song.

Рассмотрим дерево вывода ответа на запрос:

?-write('Я '),song.

Вывод:

Я бегу, бегу, бегу, бегу, бегу, бегу, бегу, бегу, …

ПРОГРАММА 2

song(X):-(X>1),write('бегу,'),(Y is X - 1),song(Y).

song(1).

?-write('Я '),song(3),write('бегу по гаревой дорожке').

Вывод:

Я бегу, бегу, бегу по гаревой дорожке

true

**Задание 1.2**

Запустите программу с отладчиком (переход в режим отладки производится с помошью предиката debug). Не забудьте установить точки остановы. Введите запросы:

?-factorial(3,X).

?-factorial(1,X).

?-factorial(0,X).

?-factorial(3,X).

Вывод:

X = 6

false

?-factorial(1,X).

Вывод:

X = 1

false

?-factorial(0,X).

false

**Задание 1.3**

Выполните программу 4 в режиме трассировки. Введите запрос:

?-factorial(3,F).

Программа 4

factorial(N,FactN):- fact(N,FactN,1,1).

fact(N,FactN,I,P):- /\* накопитель I - аналог счетчика \*/

I<N, /\* накопитель P – промежуточное значение факториала\*/

I1 is I+1, /\* - значение факториала \*/

P1 is P\*I1,

fact(N,FactN,I1,P1).

fact(N,FactN,N,FactN).

Вывод

F = 6

false

**Задание 1.4**

Выполните программу 5 в режиме трассировки. Введите запрос и нарисуйте для него дерево вывода:

?-factorial(4,F).

ПРОГРАММА 5.

Итеративное определение факториала (вариант 2, более эффективный).

factorial(N,FactN):- fact(N,FactN,1).

fact(N,FactN,P):-

N>0,

P1 is P\*N,

N1 is N-1,

fact(N1,FactN,P1).

fact(0,FactN,FactN).

?-factorial(4,F).

Вывод:

F = 24

**Задание 1.5**

Запустите программу 6, введите запросы:

?-kurs(1,X).

?-kurs(N,gruppa(X,Y)).

?-kurs(N1,gruppa(X, gruppa(Y,Z))).

ПРОГРАММА 6.

Рекурсивное определение списка студентов.

/\*группа номер 1,состоящая из студентов 'Шекспир','Мольер','Чехов'\*/

kurs(1,gruppa('Шекспир',gruppa('Мольер',gruppa('Чехов',empty)))).

kurs(2,gruppa('Гильберт',gruppa('Эйлер',gruppa('Лейбниц',

gruppa('Кантор',empty))))).

Вывод:

?-kurs(1,X).

X = gruppa('Шекспир', gruppa('Мольер', gruppa('Чехов', empty)))

?-kurs(N,gruppa(X,Y)).

N = 1, X = 'Шекспир', Y = gruppa('Мольер', gruppa('Чехов', empty))

N = 2, X = 'Гильберт', Y = gruppa('Эйлер', gruppa('Лейбниц', gruppa('Кантор', empty)))

?-kurs(N1,gruppa(X, gruppa(Y,Z))).

N1 = 1, X = 'Шекспир', Y = 'Мольер', Z = gruppa('Чехов', empty)

N1 = 2, X = 'Гильберт', Y = 'Эйлер', Z = gruppa('Лейбниц', gruppa('Кантор', empty))

**Задание 1.6**

Напишите программу для определения n-го числа Фибоначчи без накопителей и с накопителями. Числа Фибоначчи определяются следующим законом: первые два числа равны единице, а каждое последующее равно сумме двух предыдущих. То есть получим ряд 1,1,2,3,5,8,13,21,34,…

fib(N,X) :-

fib1(N,1,1,0,X).

fib1(N,N,X,Y,X) :-!.

fib1(N,M,X,Y,F) :-

M1 is M+1,

X1 is X+Y,

Y1 is X,

fib1(N,M1,X1,Y1,F).

Вывод:

?-fib(1,X).

X = 1

?-fib(2,X).

X = 1

?-fib(3,X).

X = 2

?-fib(4,X).

X = 3

?-fib(10,X).

X = 55

**Задание 1.7**

Напишите программу для определения суммы нечетных (четных) чисел из n первых чисел Фибоначчи.

fib2(N, Fib):-

fib2(N, Fib, 1, 1, 0).

fib2(N, Fib, I, Previous1, Previous2):- I<N,

I1 is I+1,

Fib1 is Previous1+Previous2,

fib2(N, Fib, I1, Fib1, Previous1).

fib2(N, Fib, N, Fib, \_).

chet(X):-

F is X mod 2,

F == 0.

chet\_sum(N, A):-

chet\_sum(N, A, 0, 0).

chet\_sum(N, A, I, Sum):-

I<N,

I1 is I + 1,

fib2(I1, FibNumber),

( chet(FibNumber) -> Sum1 is Sum + FibNumber ; Sum1 is Sum ),

chet\_sum(N, A, I1, Sum1).

chet\_sum(N, A, N, A).

Запрос:

?-chet\_sum(5,N).

N = 2

false

?-chet\_sum(6, N).

N = 10

false

**Задание 1.8**

Дано число. Проверить, является ли оно суммой нечетных (четных) чисел из n первых чисел Фибоначчи. Найтичисло n.

fib2(N, Fib):-

fib2(N, Fib, 1, 1, 0).

fib2(N, Fib, I, Previous1, Previous2):- I<N,

I1 is I+1,

Fib1 is Previous1+Previous2,

fib2(N, Fib, I1, Fib1, Previous1).

fib2(N, Fib, N, Fib, \_).

chet(X):- F is X mod 2,

F == 0.

chet\_sum(N, A):-

chet\_sum(N, A, 0, 0).

chet\_sum(N, A, I, Sum):- I<N,

I1 is I + 1,

fib2(I1, FibNumber),

( chet(FibNumber) -> Sum1 is Sum + FibNumber ; Sum1 is Sum ),

chet\_sum(N, A, I1, Sum1).

chet\_sum(N, A, N, A).

check\_num(Num, N):-

check\_num(Num, N, 0, 0).

check\_num(Number, N, Ans, I):-

Ans =< Number,

I1 is I+1,

chet\_sum(I1, Ans1),

( Ans1 == Number -> Ans2 is Number+1,

write('Да, это число равно сумме четных чисел до n-го числа Фибоначчи');

Ans2 is Ans1 ),

check\_num(Number, N, Ans2, I1).

check\_num(\_, N, \_, N).

Запрос:

?-check\_num(3, N).

N = 6

N = 5

N = 4

N = 3

N = 2

N = 1

N = 0

?-check\_num(2, N).

Да, это число равно сумме четных чисел до n-го числа Фибоначчи

N = 3

N = 2

N = 1

N = 0

?-check\_num(10, N).

Да, это число равно сумме четных чисел до n-го числа Фибоначчи

N = 6

N = 5

N = 4

N = 3

N = 2

N = 1

N = 0

**Задание 1.9**

Напишите программу для вычисления функции Аккермана, определенной на множестве пар неотрицательных чисел. 𝐴(𝑋,𝑌)={𝑌+1, при 𝑋=0𝐴(X-1,1), при 𝑋>0,𝑌=0𝐴(𝑋−1,𝐴(𝑋,𝑌−1)), при 𝑋>0,𝑌>0

ak(0, N, Val):-

Val is N+1.

ak(M,0,Val) :-M > 0,

M1 is M-1,

ak(M1,1,Val).

ak(M,N,Val) :-M > 0,N > 0,

M1 is M-1,

N1 is N-1,

ak(M,N1,Val1),

ak(M1,Val1,Val).

Вывод:

?-ak(0,0,X).

X=1

false

?-ak(0,1,X).

X = 2

false

?-ak(3,2,X).

X = 29

false

?-ak(2,3,X)

X = 9

false

# **Лабораторнаяработа № 3**

## **Исходный код**

a:- write(1).

a:- write(2).

b(X):- a,X='еще'.

c:- a.

d:- a,fail.

size(1,R):- R >= 158, R < 164.

size(2,R):- R >= 164, R < 170.

size(3,R):- R >= 170, R < 176.

size(4,R):- R >= 176, R < 182.

size(5,R):- R >= 182, R < 188.

/\*\* <examples> Your example queries go here, e.g.

?-b(X).

?-c.

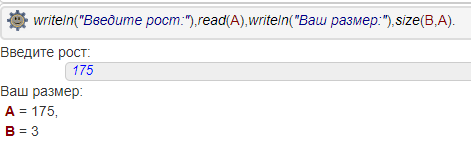
?-d.

?- repeat,a,fail.

?- writeln("Введите рост:"),read(A),writeln("Ваш размер:"),size(B,A).

\*/

Определение размера одежды по росту



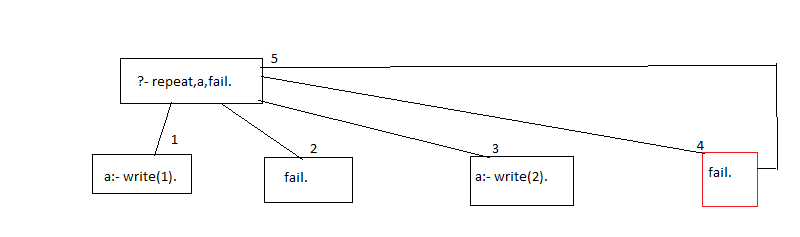
Задание 2

Выполните программу 3 с запросом

?- repeat,a,fail.

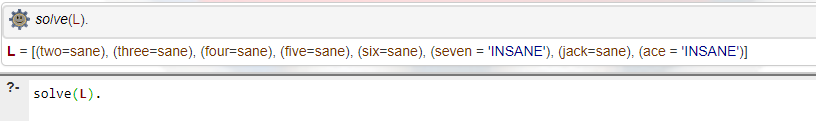
Постройте дерево вывода и объясните результат. 

Происходит вызов предиката «а», который вызывает функцию write(1), после которого вызывается предикат «fail», затем еще раз предикат «а». Вызывается уже функция write(2), после него предикат «fail». Все это повторяется бесконечно.



Задание 3.

Формализуйте и решите более сложную задачу: Тройка думает, что Туз не в своем уме. Четверка думает, что Тройка и Двойка оба не могут быть не в своем уме. Пятерка думает, что Туз и Четверка либо оба не в своем уме, либо оба в здравом рассудке. Шестерка думает, что Туз и Двойка оба в здравом уме. Семерка думает, что Пятерка не в своем уме. Что же касается Валета Червей, то он думает, что Шестерка и Семерка не оба не в своем уме. В своем ли уме Валет Червей?



state('sane').

state('INSANE').

mind(1,L):-member(ace='INSANE',L).

mind(2,L):-member(three='sane',L);member(two='sane',L).

mind(3,L):-member(ace='sane',L);member(four='sane',L);member(ace='INSANE',L);member(four='INSANE',L).

mind(4,L):-member(ace='sane',L);member(two='sane',L).

mind(5,L):-member(five='INSANE',L).

mind(6,L):-member(six='sane',L);member(seven='INSANE',L);member(six='INSANE',L);member(seven='sane',L);member(six='sane',L);member(seven='sane',L).

check\_mind('sane', N, L):-mind(N, L).

check\_mind('INSANE', N, L):-not(mind(N, L)).

solve(L):-

L=[two=A, three=B, four=C, five=D, six=E, seven=F, jack=G, ace=H],

state(A),state(B),state(C),state(D),state(E),state(F),state(G),state(H),

check\_mind(B,1,L),

check\_mind(C,2,L),

check\_mind(D,3,L),

check\_mind(E,4,L),

check\_mind(F,5,L),

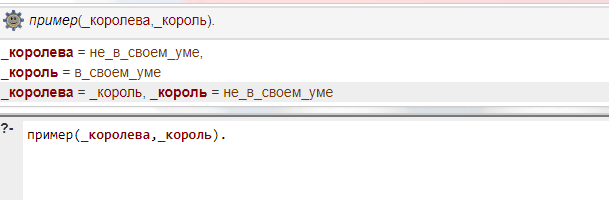
check\_mind(G,6,L).

Валет в своем уме.

Задание 4.

Используя операторную запись (нотацию) предикатов думает/2 и участник/1, измените программу о Короле и Королеве Пик так, чтобы высказывания записывались "почти" на русском языке. И правильно воспринимали такую программу: ?-op(50,fx,участник). ?-op(1000,xfy,думает\_что). участник в\_своем\_уме. участник не\_в\_своем\_уме. \_x думает\_что\_у:-(\_х=в\_своем\_уме,call(\_у)). \_x думает\_что\_у:-(\_х=не\_в\_своем\_уме,not(call(\_у))). пример0(\_Король,\_Королева):- участник \_Король, участник \_Королева, \_Король думает\_что \_Королева думает\_что \_Король=не\_в\_своем\_уме.

?-пример0(\_Король,\_Королева). Указание: при задании приоритета для операции думает\_что необходимо учитывать, что операция равно должна вычисляться ранее чем операция думает\_что.



участник(в\_своем\_уме).

участник(не\_в\_своем\_уме).

думает(\_х,\_у):-(\_х=в\_своем\_уме,call(\_у)).

думает(\_х,\_у):-(\_х=не\_в\_своем\_уме,not(call(\_у))).

пример(\_королева,\_король):-

участник(\_королева),

участник(\_король),

думает(\_королева,думает(\_король,думает(\_королева,думает(\_король,\_королева=не\_в\_своем\_уме)))).

# **Лабораторная работа № 4**

## **Исходный код**

write\_list([]).

write\_list([H|T]):- /\* разделение списка на голову и хвост, \*/

write(H), nl, /\* печать головы, пропуск строки \*/

write\_list(T). /\*рекурсивный вызов предиката от оставшегося списка\*/

length\_of([], 0).

length\_of([\_|T], L):-

length\_of(T, TailLength),

L = TailLength + 1.

member(Name,[Name |\_ ]) .

member(Name,[\_ |Tail]):-

member(Name,Tail).

list\_sum([Item], Item).

list\_sum([Item1,Item2 | Tail], Total) :-

list\_sum([Item1+Item2|Tail], Total).

list\_min([L|Ls], Min) :-

list\_min(Ls, L, Min).

list\_min([], Min, Min).

list\_min([L|Ls], Min0, Min) :-

Min1 is min(L, Min0),

list\_min(Ls, Min1, Min).

my\_max([], R, R).

my\_max([X|Xs], WK, R):- X > WK, my\_max(Xs, X, R).

my\_max([X|Xs], WK, R):- X =< WK, my\_max(Xs, WK, R).

my\_max([X|Xs], R):-my\_max(Xs, X, R).

/\*\* <examples> Your example queries go here, e.g.

?- member(3,[1,2,a,3,4,5]).

?-write\_list([1,2,a,3]).

?-length\_of([1,2,a,3], L).

?-list\_sum([1,2,0,5,6], Sum).

?-list\_min([2,4,1,6],Min).

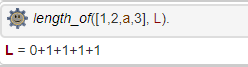
?-my\_max([1,4,8,7],Max).

\*/

Определение принадлежности элемента к списку



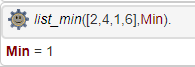
Определение длины списка



Поиск суммы элементов числового списка



Поиск минимального элемента числового списка



Поиск максимального элемента числового списка

