Пролог-процесори

Пролог-процесори. Огляд особливостей (1/2)

Prolog. Операційна семантика

Крок виконання.

Нехай у поточному запиті

$$:- C_1, C_2, \ldots, C_m.$$

для деякого C_i та лівої частини B_0 деякого правила

$$B_0 :- \boxed{B_1, B_2, \ldots, B_n}.$$

існує найбільш загальний уніфікатор (НЗУ) σ, тоді можна отримати (звертаємо увагу, що тут криється можлива недетермінованість кроку виконання) новий запит:

$$:- \sigma C_1, \sigma C_2, \ldots, \sigma C_{i-1}, \boxed{\sigma B_1, \sigma B_2, \ldots, \sigma B_n}, \sigma C_{i+1}, \ldots, \sigma C_m.$$

Детермінізм обчислень:

- послідовний перегляд атомів запиту;
- послідовний перегляд (перебір) правил;
- реалізація відкатів (*backtracking*).

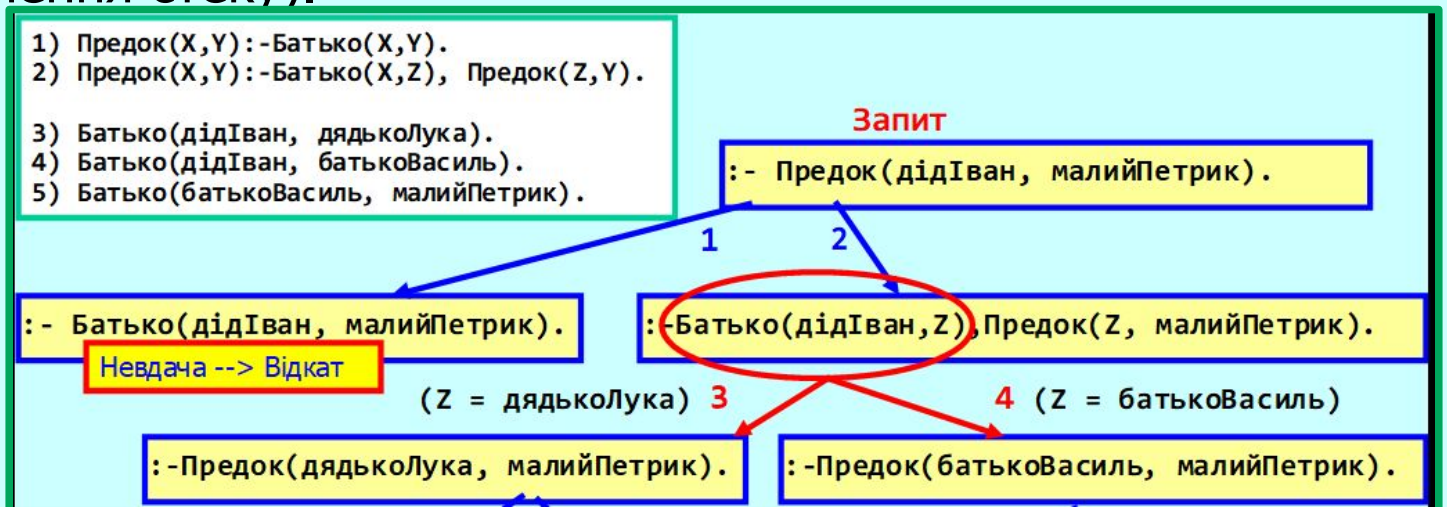
Пролог-процесори. Огляд особливостей (2/2)

Детермінізм обчислень:

- підстановки (правила) застосовуються до найлівішого атому у запиті (такий принцип спряжений зі стратегією обходу дерева “вглиб зліва-направо”);
- при наявності кількох альтернатив порядок їх застосування визначається порядком входження у текст програми:
  - у разі ***невдачі*** застосування поточної альтернативи має обиратись наступна;
  - у разі отримання невдачі для усіх альтернатив доводиться робити ***відкат*** (повернення до попереднього кроку із відновленням стану).

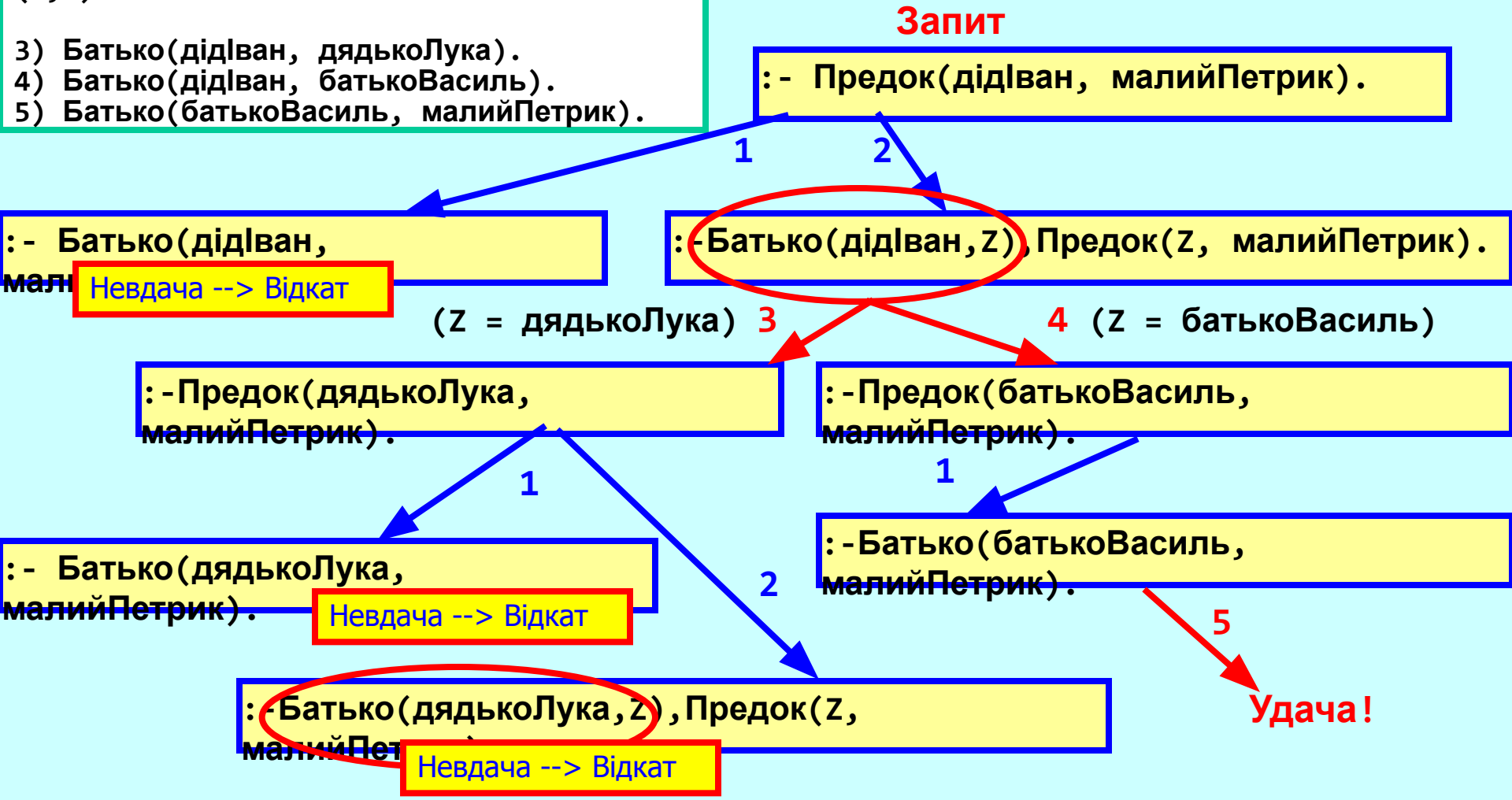
Пролог-процесори мають забезпечувати реалізацію ***відкатів***.

Увага! Можуть бути “зациклення” (з’являється нескінчена гілка)! (Ефект – переповнення стеку).



Пролог-процесори.
Ілюстрація до виконання програм

```
1) Предок(X,Y):-Батько(X,Y).
2) Предок(X,Y):-Батько(X,Z), Предок
(Z,Y).

3) Батько(дідІван, дядькоЛука).
4) Батько(дідІван, батькоВасиль).
5) Батько(батькоВасиль, малийПетрик).
```



Пролог-процесори. Приклад зациклення (1/2)

```
1¹) Предок(X,Y):- Предок(X,Z),Батько
(Z,Y).
2¹) Предок(X,Y):- Батько(X,Y).

3) Батько(дідІван, дядькоЛука).
4) Батько(дідІван, батькоВасиль).
5) Батько(батькоВасиль, малийПетрик).
```

```
Було:
1) Предок(X,Y):-Батько(X,Y).
2) Предок(X,Y):-Батько(X,Z), Предок(Z,Y).
```

```
:- Предок(дідІван, малийПетрик).
```

1

```
:-Предок(дідІван,Z), Батько(Z, малийПетрик).
```

1

```
:-Предок(дідІван,Z1), Батько(Z1, Z), Батько(Z, малийПетрик).
```

1

. . .

Пролог-процесори. Приклад зациклення (2/2)

```
ff(X,Y):-ff(X,Z),f(Z,Y).    %forefather, ancestor
ff(X,Y):-f(X,Y).
f('Іван','Петро').
f('Петро','Лука').
```

SWI-Prolog (Multi-threaded, version 6.0.0)

File Edit Settings Run Debug Help

```
% c:/Users/1/Documents/Prolog/ff.pl compil
ed 0.00 sec, 6 clauses
1 ?- ff(X,'Лука').
ERROR: Out of local stack
   Exception: (1,675,218) ff(_G126, _G212)
 ?
```

Управління відкатом

Стандартні предикати:

- *cut* (є скорочена форма “!”) – відсікання (або закріплення вибору);
- *fail* – предикат, спряжений із примусовим запуском відкату.

До техніки *Prolog*-програмування.
Пошук множин розв’язків

SWI-Prolog

```
?- append(X,Y, [a,b,c]).
```

SWI-Prolog (AMD64, Multi-threaded, version 7.6...

File Edit Settings Run Debug Help

```
?- append(X,Y, [a,b,c]).
X = [],
Y = [a, b, c] |
```

SWI-Prolog (AMD64, Multi-threaded, version 7.6...

File Edit Settings Run Debug Help

```
X = [],
Y = [a, b, c] ;
X = [a],
Y = [b, c] |
```

SWI-Prolog (AMD64, Multi-threaded, versio

File Edit Settin

```
X = [a],
Y = [b, c] ;
X = [a, b],
Y = [c] |
```

SWI-Prolog (AMD64, M

File Edit Settings Run

```
X = [a, b],
Y = [c] ;
X = [a, b, c],
Y = [] |
```

SWI-Prolog (AMD...

File Edit Settings Run Debug Help

```
?- append(X,Y, [a,b,c]).
X = [],
Y = [a, b, c] ;
X = [a],
Y = [b, c] ;
X = [a, b],
Y = [c] ;
X = [a, b, c],
Y = [] ;
false.

?- |
```

Натискання клавіші “Space” дозволяє отримувати черговий розв’язок (при його наявності).

Пошук усіх розв'язків із використанням предикату *fail*

```
1) Предок(X,Y):-Батько(X,Y).
2) Предок(X,Y):-Батько(X,Z), Предок
(Z,Y).

3) Батько(дідІван, дядькоЛука).
4) Батько(дідІван, батькоВасиль).
5) Батько(батькоВасиль, малийПетрик).
```

Предикат *fail* зручно використовувати при потребі знайти не один, а усі розв'язки деякої задачі.

Приклад. Запит для знаходження усіх синів діда Івана.

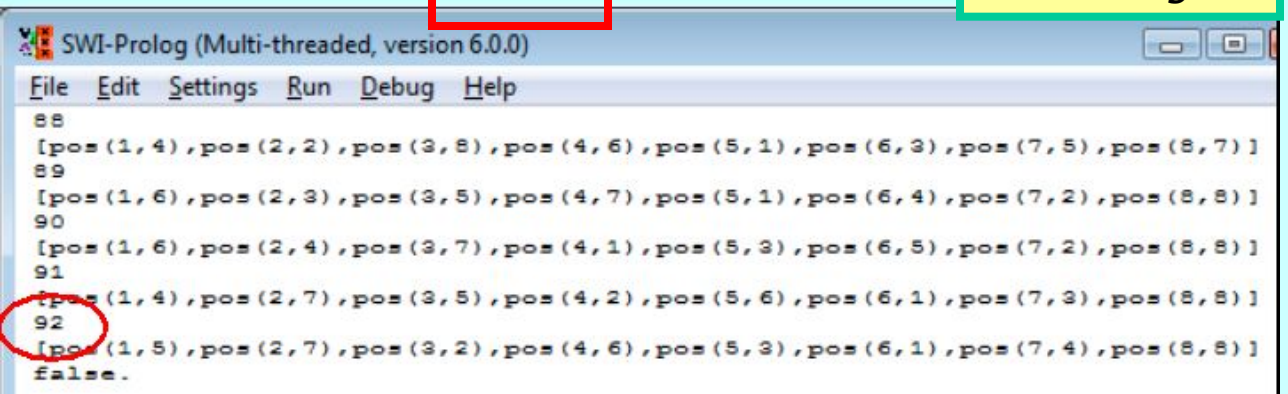
Стандартні предикати

```
Батько(дідІван, X), nl, write(X), fail.
```

Приклад 4.
Розстановка 8 ферзів на дошці 8×8 (2/2)

```
qq(L):-assertz(count(0)), pp(L), rr(L), count(Temp),
       retract(count(_)), NewTemp is Temp+1,
       assertz(count(NewTemp)), writeln(NewTemp),
       writeln(L), fail.
```

SWI-Prolog



Пошук усіх розв’язків із використанням предикату *fail*

```
?- append(X,Y,[a,b,c]),nl,write(X),write(Y),fail.
```

SWI-Prolog (AMD64, Multi-threaded, version 7.6.4)

File Edit Settings Run Debug Help

```
?- append(X,Y, [a,b,c]), nl, write(X),write(Y), fail.

[][a,b,c]
[a][b,c]
[a,b][c]
[a,b,c][]
false.
```

До техніки *Prolog*-програмування.
Скорочувані правила

Скорочувані правила (1/3)

```
my_max(A,B,A):-A>=B.
my_max(A,B,B):-A<B.
```

SWI-Prolog

Чи потрібна перевірка **X<Y** ?

```
my_max(A,B,A):-A>=B.
my_max(A,B,B).
```

Але спробуємо скористатись предикатом ***fail*** за рецептом пошуку “усіх” можливих розв’язків.

Запит:

```
my_max(3,2,Z), write(Z), nl, fail.
```

Який буде результат роботи програми?

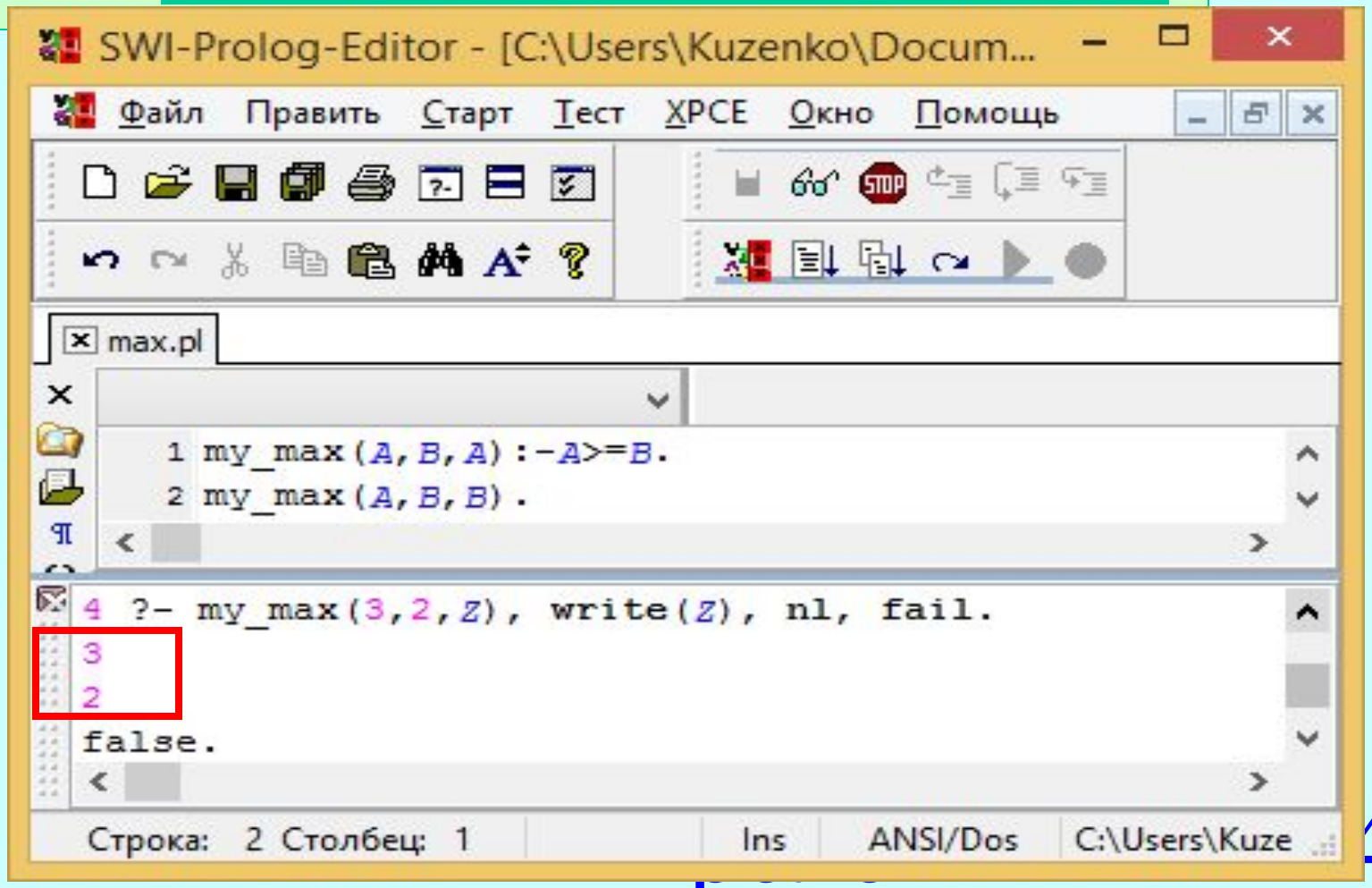
Скорочувані правила (2/3)

```
my_max(A,B,A):-A>=B.
my_max(A,B,B).
```

SWI-Prolog

Запит:

```
my_max(3,2,Z), nl, write(Z), fail.
```



Скорочувані правила та відсікання *cut* (!) (3/3)

```
my_max(A,B,A):-A>=B.
my_max(A,B,B).
```

Запит:

```
my_max(3,2,Z), write(Z), nl, fail.
```

Як же підправити програму?

```
my_max(A,B,A):-A>=B, ! .
my_max(A,B,B).
```

"Закріплення вибору".

```
member(X,[X|T]).
member(X,[Y|T]) :- X=\=Y, member(X,T).
```

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

Комбінація “!, fail”. (*SWI-Prolog*)

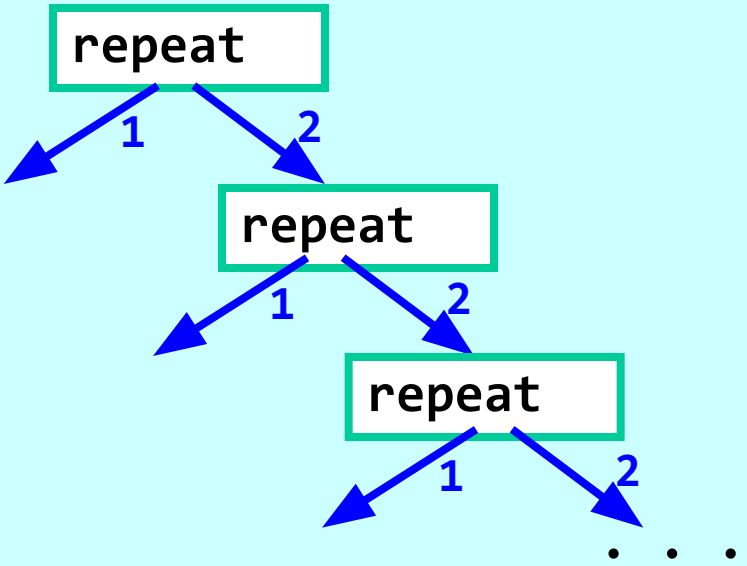
Використовується для припинення обчислень для заданої “гілки” (часто після перевірки даних та визначення їх не коректними).

Приклад.

```
check_cost(X):- X =< 0, write(“не коректна ціна”), !, fail.
```

Моделювання циклів (повтори). Предикат *repeat*

```
repeat.
repeat :- repeat.
```

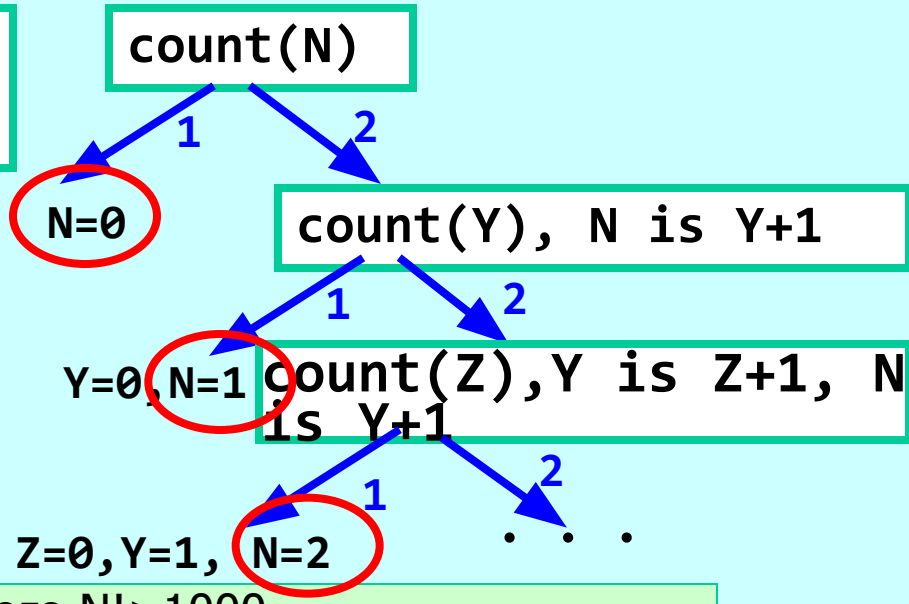


Цей предикат дозволяє відкат “розвернути” (запустити черговий крок циклічних обчислень).

```
check_stop(“стоп”).

echo:-repeat, readln(X), write(X), check_stop(X).
```

Повтори з лічильником. (*SWI-Prolog*)

```
count(0).
count(X) :- count(Y), X is Y+1.
```



Знайти найменше ціле число N, для якого N!>1000.

```
fact(0,1).
fact(X,Y) :- X>0, X1 is X-1, fact(X1,Y1), Y is Y1*X.

goal count(N), fact(N,Y), Y>1000, !, write(N).
```

```
5 ?- count(N), fact(N,Y), Y>1000, !, write(N).
7
N = 7,
Y = 5040.
```

Пошук (перевірка) елемента за його позицією у списку

```
member(X,[X|T]).
member(X,[Y|T]) :- X=\=Y, member(X,T).
```

```
pos(1, [H | _], H).
pos(N, [_ | T], Elem) :- N > 1, K is N-1, pos(K, T, Elem).
```

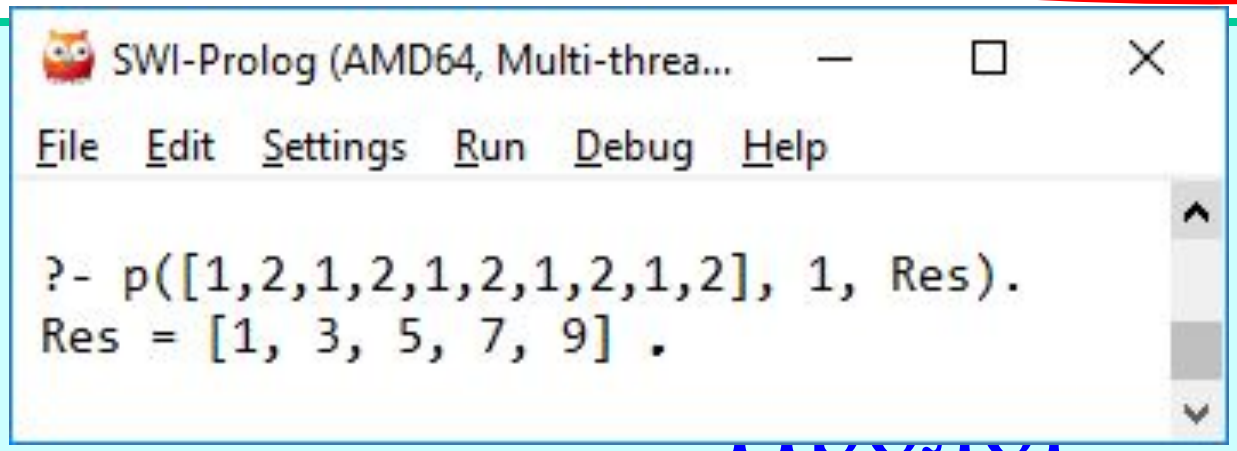
```
?- pos(3,[1,2,3],X).
X = 3 .

?- pos(6,[1,2,3],X).
false.
```

```
% 8. Той, хто мешкає в центральному будинку, п'є молоко.

H=[_,_,house(_,_,_,_,milk,_),_,_]

pos(3, H, house(_,_,_,_,milk,_))     % ще один варіант умови
```

Приклад 1 із пошуком позицій елементів списку. *SWI-Prolog*

Знайти список позицій входжень елемента *X* у список *L*.

```
% L – список, X – елемент, LRes – шуканий список
p(L,X,LRes):- positions1(L,X,LRes,1).
% 1 - лічильник C (позиція поточної голови у початковому списку)
% positions1(L,X,LRes,C).
positions1([],_,[],_).
positions1([H|T],X,L,Pos) :- X=\=H,!,Y is Pos+1,
                                positions1(T,X,L,Y).
positions1([H|T],X,[Pos|L],Pos):-X=H, Y is Pos+1, positions1(T,X,L,Y).
```



Приклад 2 із пошуком позицій елементів списку (1/2)

Знайти у списку *L* перший (найлівіший) елемент, більший за *X*, забезпечуючи пошук як значення, так і його позицію у списку.

```
find(L,X,Zn,PosZn) % L - список, X - задане значення,
                   % Zn - шуканий елемент та PosZn - його позиція

find(L,X,Zn,PosZn):- find1(L,X,Zn,PosZn,1).     %SWI-Prolog
% 1 - лічильник C(позиція поточної голови у початковому списку)

find1([],X,_,0,_).

find1([H|T],X,H,P,P):-H>X,!.

find1([H|T],X,Zn,PosZn,P):-H<=X, Y is Pos+1, find1(T,X,Zn,PosZn,Y).
```

Ознака відсутності шуканого елемента!

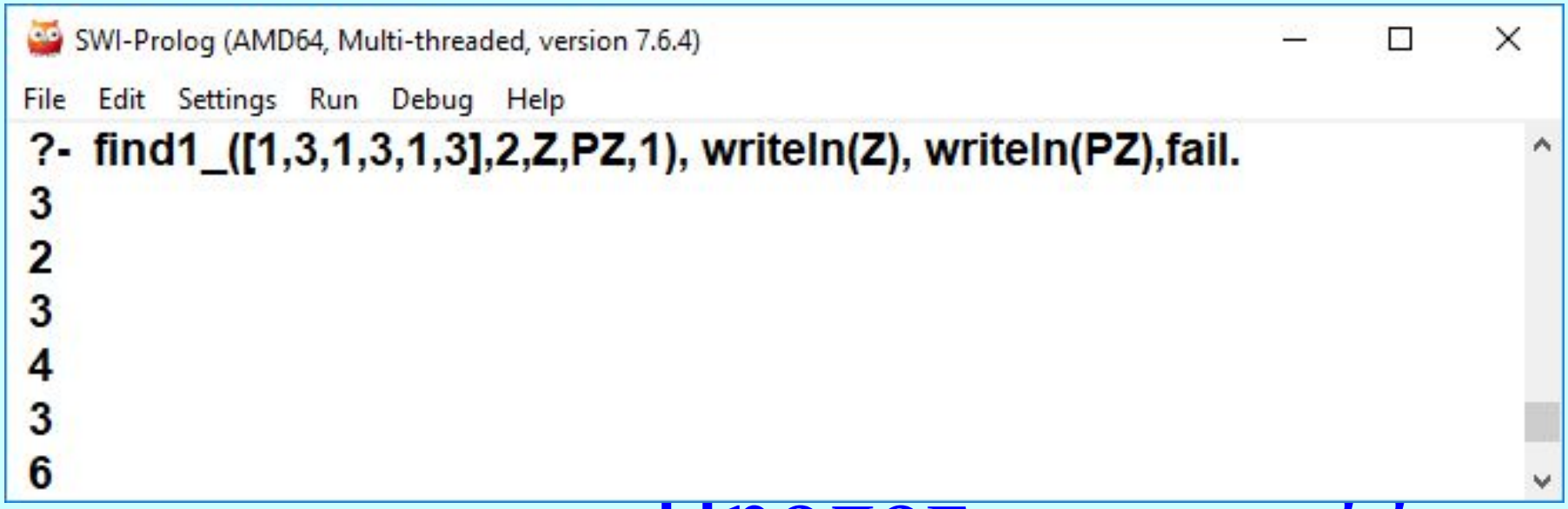
```
?- find([1,3,1,3,1,3],2,Z,PZ).
Z = 3,
PZ = 2.

?- find([1,3,1,3,1,3],2,Z,PZ), writeln(Z), writeln(PZ),fail.
3
2
false.
```

Приклад 2 із пошуком позицій елементів списку (2/2)

```
find1([H|T],X,H,P,P):-H>X,!.
find1([H|T],X,Zn,PosZn,P):-H<=X, Y is Pos+1, find1(T,X,Zn,PosZn,Y).
```

Перестрахувались навіть двічі!

```
find1_([H|T],X,H,P,P):-H>X.
find1_([H|T],X,Zn,PosZn,P):-Y is Pos+1, find1_(T,X,Zn,PosZn,Y).
```



Приклад 3 із пошуком позицій елементів списку

Знайти у списку *L* останній (найправіший) з елементів, більших за *X*, забезпечуючи пошук як значення, так і його позицію у списку.

```
findLast(L,X,Zn,PosZn):- findLast1(L,X,Zn,PosZn,1).

findLast1([],X,_,0,_).

findLast1([H|T],X,Zn,PosZn,P):-H<=X, P1 is P+1,

                                  findLast1(T,X,Zn,PosZn,P1).

findLast1([H|T],X,H,P,P):-H>X,find(T,X,_,Z),Z=0.!

findLast1([H|T],X,Zn,PosZn,P):-H>X,find(T,X,_,Z),Z<>0,

 P1 is P+1, findLast1(T,X,Zn,PosZn,P1).
```

Ознака відсутності шуканого елемента!

Ознака відсутності шуканого елемента!

З попередньої задачі!

```
?- findLast([1,2,3,1,2,3,1,2,3],2,Zn,PosZn).
Zn = 3,
PosZn = 9 .
```

Приклад 4 (1/3)

Повставляти елементи одного списку у впорядкований за зростанням другий список. Сформувати список із номерами позицій, які займають вставлені елементи в новому списку. (Задача на зразок 50).

```
bubl(L,LSort) :-trans(L,L1),!,bubl(L1,LSort).
bubl(L,L).
trans([X,Y|Tail], [Y,X|Tail]) :- greater(X,Y).
trans([X|T], [X|T1]) :- trans(T,T1).

greater([X,_],[Y,_]):-X>Y.

cod([],[],_).  % для "розмітки" списків (за 3 параметром):
               %  1 - для першого списку, 2 - для другого
cod([H|T],[[H,X]|RT],X):-cod(T,RT,X).

decod([],[]).
decod([[H,_]|T],[H|RT]):-decod(T,RT).

append([],L,L).
append([H|T],L,[H|R]):-append(T,L,R).

res(L,LR):-res1(L,LR,1).   % вибір елементів з міткою 1
                           % (з бувшого 1-го списку) та фіксація їх позицій
res1([],[],_).
res1([[_,2]|T],R,P):-P1 is P+1, res1(T,R,P1).
res1([[X,1]|T],[[X,P]|R],P):-P1 is P+1, res1(T,R,P1),.
```

Приклад 4 (2/3)

Повставляти елементи одного списку у впорядкований за зростанням другий список. Сформувати список із номерами позицій, які займають вставлені елементи в новому списку.

```
query(L1,L2,R1Sort,R2):-
    cod(L1,LL1,1),
    cod(L2,LL2,2),
    append(LL1,LL2,LL),
    bubl(LL,LLS),
    decod(LLS, R1Sort),
    res(LLS,R2).

% test:     query([7,3,5,1,9],[2,4,6,8,10],R1Sort,R2).
```

```
    writeln(LL1),
    writeln(LL2),
    writeln(LL),
    writeln(LLS),
    writeln(R1Sort),
```

Можна закоментувати

```
6 ?- query([7,3,5,1,9],[2,4,6,8,10],R1Sort,R2).
[[7,1],[3,1],[5,1],[1,1],[9,1]]
[[2,2],[4,2],[6,2],[8,2],[10,2]]
[[7,1],[3,1],[5,1],[1,1],[9,1],[2,2],[4,2],[6,2],[8,2],[10,2]]
[[1,1],[2,2],[3,1],[4,2],[5,1],[6,2],[7,1],[8,2],[9,1],[10,2]]
[1,2,3,4,5,6,7,8,9,10]
R1Sort = [1, 2, 3, 4, 5, 6, 7, 8, 9|...],
R2 = [[1, 1], [3, 3], [5, 5], [7, 7], [9, 9]]
```

Приклад 4 (3/3)

SWI-Prolog-Editor - [C:\Users\Kuzenko\Documents\Prolog\task4_...

Файл Править Старт Тест XPCE Окно Помощь

task4_v3.pl

```
21
22 query(L1,L2,R1Sort,R2):-
23         cod(L1,LL1,1),          writeln(LL1),
24         cod(L2,LL2,2),          writeln(LL2),
25         append_my(LL1,LL2,LL),  writeln(LL),
26         bubl(LL,LLS),           writeln(LLS),
27         decod(LLS, R1Sort),     writeln(R1Sort),
28         res(LLS,R2).
```

```
4 ?- query([7,3,5,1,9],[2,4,6,8,10],R1Sort,R2).
[[7,1],[3,1],[5,1],[1,1],[9,1]]
[[2,2],[4,2],[6,2],[8,2],[10,2]]
[[7,1],[3,1],[5,1],[1,1],[9,1],[2,2],[4,2],[6,2],[8,2],[10,2]]
[[1,1],[2,2],[3,1],[4,2],[5,1],[6,2],[7,1],[8,2],[9,1],[10,2]]
[1,2,3,4,5,6,7,8,9,10]
R1Sort = [1, 2, 3, 4, 5, 6, 7, 8, 9|...],
R2 = [[1, 1], [3, 3], [5, 5], [7, 7], [9, 9]]
```

Строка: 20 Столбец: 30 Ins ANSI/Dos C:\Users\Kuzenko\Documents\Prolog

Повставляти елементи одного списку у впорядкований за зростанням другий список. Сформувати список із номерами позицій, які займають вставлені елементи в новому списку.

Приклад. Перевертання списку

```
inv([],[]).   % 1 -  вх список, 2 - рез список
inv(L_in,L_out):-inv1(L_in,L_out,[]).
inv1([],L,L).  % 1 -  вх список, 3 - список-додаток,
   % 2 - рез список
inv1([H|T],X,Y):-inv1(T,X,[H|Y]).
```

Додаток 1

fib, isFib

```
fib(1,0).
fib(2,1).
fib(N,F) :- N>2, N2 is N-2, N1 is N-1,
            fib(N2,F2), fib(N1,F1), F is F1+F2.

count(0).
count(X) :- count(Y), X is Y+1.

isFib(N) :- N>=0, count(X), fib(X,F), F>=N, !, F=:=N.
```

isFib

SWI-Prolog-Editor - [C:\Users\Kuzenko\Documents\Prolog\2017_fib_prime.pl]

File Edit Start Test XPCE Window Help

2017_fib_prime.pl

```
8  isFib(N) :- N>=0, count(X), fib(X,F), F>=N, !, F=:=N.
9
10 isFibErr(N) :- N>=0, count(X), fib(X,F), F>=N, F=:=N.
11
```

```
?- isFib(5).
true.

?- isFib(10).
false.

?- isFibErr(10).
ERROR: Out of local stack
?-
```

Line: 10 Column: 61 | Ins | ANSI/Dos | C:\Users\Kuzenko\Documents\Prolog\2017_fib_prime.

genToN1, isPrime

```
% genToN1(N, [2,3, ..., N-1]) - true, N>3
genToN1(2,[]).
genToN1(N, [N1|T]):-N>2, N1 is N-1, genToN1(N1,T).

% checkDiv(N,Lst)=true  if N mod Xi =\= 0 for all Xi from Lst
checkDiv(N, []).
checkDiv(N, [H|T]) :- N mod H =\= 0, checkDiv(N,T).

isPrime(2).
isPrime(N) :- N>2, genToN1(N,L), checkDiv(N,L).
```

genToN1

SWI-Prolog-Editor - [C:\Users\Kuzenko\Documents\Prolog\...

File Edit Start Test XPCE Window Help

2017_fib_prime.pl

```
% genToN1(N, [2,3, …, N-1]) - true, N>3
genToN1(2,[]).
genToN1(N, [N1|T]):-N>2, N1 is N-1, genToN1(N1,T).
```

```
CC = [11, 10, 9, 8, 7, 6, 5, 4, 3|...] ;
false.

?- set_prolog_flag(answer_write_options,[max_depth(0)]).
true.

?- genToN1(12,CC).
CC = [11,10,9,8,7,6,5,4,3,2] ;
false.
```

Line: 7 Column: 89 | Ins | ANSI/Dos | C:\Users\Kuzenko\Docume

isPrimeByCount

```
countFrom2(2).
countFrom2(X) :- countFrom2(Y), X is Y+1.

isPrimeByCount(2).
isPrimeByCount(N) :-
                    N>2,countFrom2(X), N mod X =:= 0, !, X=:=N.
```

isPrime II

```
countFrom2(2).
countFrom2(X) :- countFrom2(Y), X is Y+1.
progon(N):- N1 is N-1, countFrom2(X), NX is (N mod X),
   (NX =:= 0 -> asserta(flag(X)); true), X=:=N1.

isPrime_II(2).
isPrime_II(N):- N>2, clear_flags, asserta(flag(0)), progon(N),
   retract(flag(F)), F=:=0.

clear_flags:-retract(flag(_)), fail.
clear_flags.
```

Додаток 2

SWI-Prolog-Editor. Завантаження та запуск

SWI-Prolog-Editor - [C:\Users\Kuzenko\Document...

Файл Править Старт Тест XPCE Окно Помощь

Новый Strg+N
Открыть... Strg+O
Открыть проект
Открыть повторно
Сохранить Strg+S
Сохранить как...
Сохранить все Strg+Alt+S
Сохранить как проект
Закрыть Strg+F4
Закрыт все

1

:-append(T,U,V).

g-Editor - [C:\Users\Kuzenko\

Старт Тест XPCE Окно

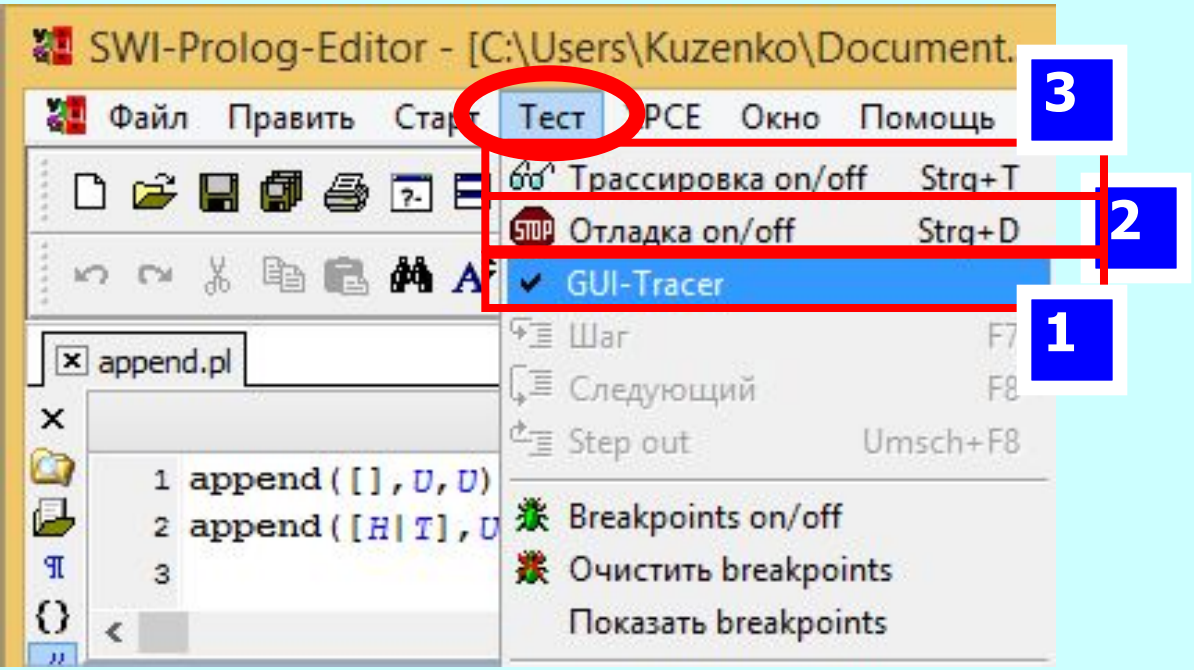
2

Запустить F9
Запустить все
Повторить запрос
Продолжить
Прекратить
Исходная папка
Перезапустить Strg+F9

append.pl

1 append([
2 append([
3

SWI-Prolog-Editor. Параметри трасування



SWI-Prolog-Editor. Виконання (із трасуванням)

SWI-Prolog-Editor - [C:\Users\Kuzenko\Document...

Файл Править Старт Тест XPCE Окно Помощь

append.pl

```
1 append([],U,U).
2 append([H|T],U,[H|V]):-append(T,U,V).
3
```

```
4 ?- debug.
true.

[debug] 5 ?- trace.
true.

[trace] 5 ?- append([a,b],[c],Z).
```

Строка: 1 Столбец: 20 Ins ANSI/Dos

Клавіша "*Enter*" розпочинає виконання, клавіша "Space" може бути використана для ініціювання чергового кроку

SWI-*Prolog-Editor*. Трасування. Додаткове вікно (1/2)

c:/users/kuzenko/documents/prolog/append.pl

Tool Edit View Compile Help

Bindings

U = [c]
H = b
T = []

Call Stack

6 append/3
7 append/3
8 append/3

```
append([],U,U).
append([H|T],U,[H|V]):-append(T,U,V).
```

Call: append/3

Tool Edit View Compile

Bindings

U = [c]
H = a
T = [b]

6 append/3
7 append/3

```
append([],U,U).
append([H|T],U,[H|V]):-append(T,U,V).
```

Call: append/3

Tool Edit View Compil

Bindings

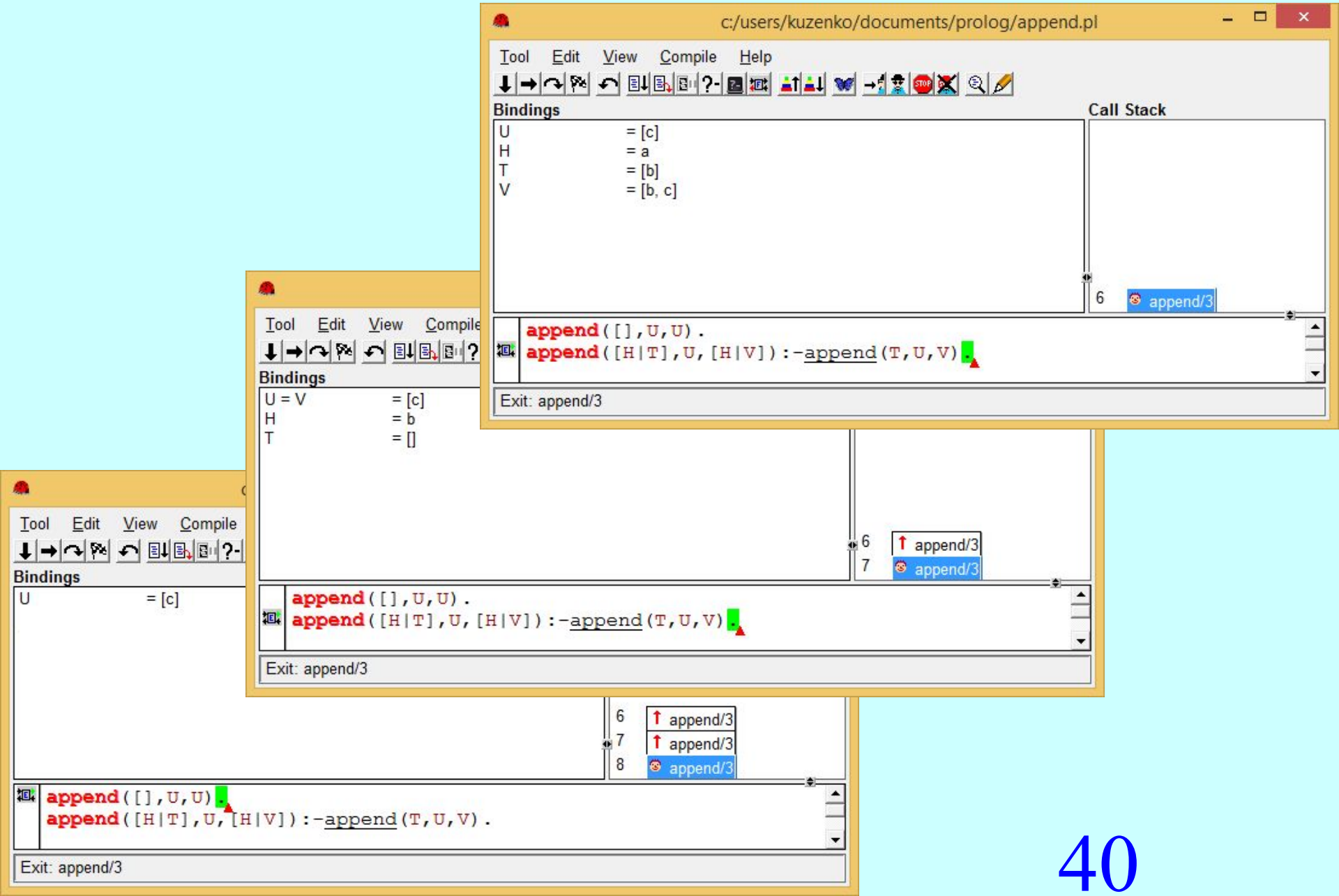
U = [c]
H = a
T = [b]

6 append/3

```
append([],U,U).
append([H|T],U,[H|V]):-append(T,U,V).
```

Черговий крок – клавіша “Space”

SWI-Prolog-Editor. Трасування. Додаткове вікно (2/2)



SWI-Prolog-Editor. Результат виконання

SWI-Prolog-Editor - [C:\Users\Kuzenko\Documents\...

Файл Править Старт Тест XPCE Окно Помощь

append.pl

```
1 append([],U,U).
2 append([H|T],U,[H|V]):-append(T,U,V).
```

```
[trace] 5 ?- append([a,b],[c],Z).
Z = [a, b, c].
```

Строка: 5 Столбец: 69 Ins ANSI/Dos

Додаток 3

Prolog. Online

Google

prolog online

Усе Відео Зображення Карти Новини Більше Інструмен

Приблизна кількість результатів: 4 530 000 (0,35 сек.)

Execute **Prolog** Script **Online**
www.compile**online**.com/execute_**prolog**_onlin... Перекласти цю сторінку
Execute **Prolog** Script **Online** - You can compile and run, execute your source code related to Java, JSP, Cold Fusion, C, C++, Pascal, Fortran, PHP, Perl, Ruby ...

prolog online - Пошук Go Execute Prolog Script Onli

www.compileonline.com/execute_prolog_online.php

Мова цієї сторінки: англійська Перекласти її? Перекласти Ні

compileonline</>com - Execute Prolog Script Online (GNU Prolog 1.4.0)

Execute Script main.pr input.txt Default Ace E

```
1 :- initialization(main).
2 main :- write('Hello World!').
3
```

Prolog. Online

prolog online - Пошук Go | Execute Prolog Script Onli

www.compileonline.com/execute_prolog_online.php

Мова цієї сторінки: англійська | Перекласти її? | Перекласти | Ні | Параметри

compileonline</>com - Execute Prolog Script Online (GNU Prolog 1.4.0)

About | Help | Web Editors

Execute Script | main.pr | input | Default Ace Editor

```
1  :- initialization(main).
2  main :- append_my([a,b],[c],Z), write('Z = '),
   write(Z).
3  append_my([],U,U).
4  append_my([H|T],U,[H|V]):-append_my(T,U,V).
5
6
```

Result | Download Files

Executing the program....

$gprolog --consult-file main.pr

```
GNU Prolog 1.4.0
By Daniel Diaz
Copyright (C) 1999-2011 Daniel Diaz
compiling /web/com/139413947330062/ma
/web/com/139413947330062/main.pr comp
Z = [a,b,c]
```

Command Line Arguments: | STDIN Input:

Prolog. Online

Execute Prolog Script Onli

www.compileonline.com/execute_prolog_online.php

Мова цієї сторінки: англійська Перекласти її? Перекласти Ні Параметри

compileonline</>com - Execute Prolog Script Online (GNU Prolog 1.4.0)

Home Languages Web Editors About Help

Execute Script main.pr input.txt Default Ace Editor

```
:- initialization(main).


bubl(L,LSort) :-trans(L,L1),!,bubl(L1,LSort).
bubl(L,L).
trans([X,Y|Tail], [Y,X|Tail]) :- greater(X,Y).
trans([X|T], [X|T1]) :- trans(T,T1).

greater([X,_],[Y,_]):-X>Y.

cod([],[],_).          % для "розмітки" списків (за 3 параметром): 1 - для першого списку, 2 - для другого
cod([H|T],[[H,X]|RT],X):-cod(T,RT,X).

decod([],[]).
decod([[H,_]|T],[H|RT]):-decod(T,RT).

append_my([],L,L).
append_my([H|T],L,[H|R]):-append_my(T,L,R).

res(L,LR):-res1(L,LR,1).   % вибір елементів з міткою 1 (з бувшого 1-го списку) та фіксація їх позицій
res1([],[],_).
res1([[_,N]|T],R,P):-P1 is P+1, res1(T,R,P1), N=2.
res1([[X,N]|T],[[X,P]|R],P):-P1 is P+1, res1(T,R,P1), N=1.

query4(L1,L2,R1Sort,R2):-
  cod(L1,LL1,1),    write(LL1), nl,
  cod(L2,LL2,2),    write(LL2), nl,
  append_my(LL1,LL2,LL),  write(LL), nl,
  bubl(LL,LLS),   write(LLS), nl,
  decod(LLS, R1Sort),   write(R1Sort), nl,
  res(LLS,R2).

main :- query4([7,3,5,1,9],[2,4,6,8,10],R1Sort,R2), write(R1Sort),
        write(R2).
```

Result Download Files

Executing the program....

$gprolog --consult-file main.pr

```
GNU Prolog 1.4.0
By Daniel Diaz
Copyright (C) 1999-2011 Daniel Diaz
compiling /web/com/139236484524666/main.pr for byte code...
/web/com/139236484524666/main.pr compiled, 38 lines read - 6302 bytes written, 11 ms
[[7,1],[3,1],[5,1],[1,1],[9,1]]
[[2,2],[4,2],[6,2],[8,2],[10,2]]
[[7,1],[3,1],[5,1],[1,1],[9,1],[2,2],[4,2],[6,2],[8,2],[10,2]]
[[1,1],[2,2],[3,1],[4,2],[5,1],[6,2],[7,1],[8,2],[9,1],[10,2]]
[1,2,3,4,5,6,7,8,9,10]
[1,2,3,4,5,6,7,8,9,10][[1,1],[3,3],[5,5],[7,7],[9,9]]
```

Command Line Arguments: STDIN Input: