```
% Ukazka prace se seznamy a nekolik veselych
komentaru,
% abyste se v tom vyznali i vecer pred
zkouskou.
% rohlik@kiv.zcu.cz


% Prirazovat seznamy je drsny:
seznam([1, 2, 3]).
pikacu1 :- write([1, 2, 3, 4, 5]).
pikacu2 :- seznam(X), write(X).


% Vzpomenme na prednasku:
member(H, [H|T]).
member(X, [H|T]):-member(X, T).


% A jeste jedna trpka vzpominka:
append([], X, X).
append([H|X], Y, [H|Z]) :- append(X, Y, Z).

% Vyzkousejte v listeneru nasledujici dotazy:

% ?- append([1,2,3],[4,5,6],LIST).
% ?- append(L1,L2,[1,2,3,4,5]).


% Upovidanejsi verze predikatru append
(funkcne vsak naprosto shodna):
append2([], X, X) :- write('Zabralo prvni
pravidlo'), nl.
append2([H|X], Y, [H|Z]) :- write('H je '),
write(H), write('   '), write(X), write(Y),
write(Z), nl, append2(X, Y, Z), write(X),
write(Y), write(Z), nl.

% Vyzkousejte v listeneru nasledujici dotaz:
% ?- append2([1,2,3],[4,5,6],LIST).
```

```
% Toto je soubor, na kterem lze demonstrovat
hledani cesty orientovanym grafem
% Dlouho po pulnoci vyrobil a veselymi
komentari opatril rohlik@kiv.zcu.cz
%
% Vsimete si mj. stabni kultury a sve
programy piste podle tohoto vzoru



% definice grafu
hrana(1, 4).
hrana(1, 5).
hrana(2, 1).
hrana(2, 4).
hrana(3, 1).
hrana(3, 2).
hrana(3, 4).
hrana(4, 5).


% rekurzivni definice, posloupnost
prochazenych uzlu se uklada do seznamu
cesta(X,Y,[X,Y]) :- hrana(X,Y).
cesta(X,Y,[X|T]) :- hrana(X,Z), cesta(Z,Y,T).


% pouziti predikatu FINDALL a vypis seznamu s
X, pro ktere je splnen predikat cesta(2, 5,
X)
fa1 :- findall(X, cesta(2, 5, X),
LIST_OF_Xs), write(LIST_OF_Xs).

% jina moznost aneb jak "udelat to same"
jinymi prostredky
fa2 :- cesta(2, 5, X), write(X), nl, fail.
fa2.
```

```
% Priklad na vysvetleni funkce predikatu "!"
% Program wumpus.pro vykuchal a co zbylo
veselymi
% komentari opatril rohlik@kiv.zcu.cz


% Graf - ktera mistnost je spojena s kterou
connected(1, 4).
connected(1, 5).
connected(2, 1).
connected(2, 4).
connected(3, 1).
connected(3, 2).
connected(3, 4).
connected(4, 5).


% Kolikaty pokus o splneni cile ... pri kazdem
neuspechu
% plneni "connected" se diky "!" snazime najit
novou unifikaci
% promene X v predikatu "pokus(X)", coz je
podcil cile "go".
pokus(1).
pokus(2).
pokus(3).
pokus(4).
pokus(5).


% Inicializace. Na zacatku programu je Wumpus
v mistnosti cislo 2
wumpus(2).


% Sledujeme pokusy a pohyb Wumpuse
go :-  pokus(X), write('Pokus cislo '),
write(X),
       nl, movewumpus, nl,
       write('Skoncil pokus cislo '),
write(X).


% Nejprve s X unifikuje s cislem mistnosti,
kde prave Wumpus je.
% Pak se hleda mistnost Y, do ktere by bylo
mozno Wumpuse presunout.
% Z databaze se vyjme fakt "wumpus(X)" a
naopak se do ni prida
% fakt "wumpus(Y)".
movewumpus:-
        wumpus(X),
       write('Predikat "movewumpus" v akci ...
'),nl,
       write('Pred testem CONNECTED je Wumpus
v mistnosti '), write(X),nl,
       !,connected(X,Y),retract(wumpus(X)),
       assert(wumpus(Y)),
       write('Po testu CONNECTED je Wumpus v
mistnosti '), write(Y).
movewumpus:-write('I do nothing').


% Teprve tady bude videt kouzlo operatoru "!"
-> po ctvrtem
% prikazu "go." se neunifikuje s X v predikatu
wumpus(X),
% ale v predikatu pokus(X) tedy "o uroven
vyse". Je to
% videt napriklad z toho, ze se podruhe
nevypisuje hlaska
% 'Predikat "movewumpus" v akci ... '
%
% Pro stale tapajici: Zkuste odstranit
vykricnik z pravidla
% "movewumpus" a uvidite.
wumpus(8).
```

```
main:-
        write($Welcome to WUMPUS HUNT$),nl,nl,
        write($Type "helpme." for help.$),nl,
        write($Type "quit." if you wimp
out.$),nl,nl,!,
        observe,go.

connected(1,2).connected(2,3).connected(3,4).
connected(4,5).connected(5,6).connected(5,7).
connected(6,19).connected(19,18).connected(18,20).
connected(20,17).connected(17,16).connected(18,16).
connected(15,14).connected(15,16).connected(14,11).
connected(11,10).connected(12,13).connected(11,12).
connected(8,2).connected(8,9).connected(9,11).
connected(8,10).

connected(2,1).connected(3,2).connected(4,3).
connected(5,4).connected(6,5).connected(7,5).
connected(19,6).connected(18,19).connected(20,18).
connected(17,20).connected(16,15).connected(16,17).
connected(16,18).connected(14,15).connected(11,14).
connected(12,11).connected(13,12).connected(10,8).
connected(10,11).connected(9,8).connected(11,9).
connected(2,8).

wumpus(17).
you(2).
arrows(5).
pit(10).
pit(13).
pit(16).
pit(7).
bats(9).
bats(4).


eaten:-
        you(X),wumpus(X),
        write($It's the WUMPUS! You lose.$),nl,end.
eaten.

fall:-
        you(X),pit(X),
        write($AA! You fall into a pit. You
lose!$),nl,end.
fall.

carried:-
        you(X),bats(X),get_room(G),check(G,R),
        write($The bats carry you to room
#$),write(G),nl,
        retract(you(X)),assert(you(G)),observe.
carried.

check(X,X):- you(X),fail.
check(X,Y):- X \= Y.
check(X,Y):- Y is X + 1, X \= 0, X \= 20.
check(X,Y):- check(X,Y).

get_room(X):-
        X is integer(random * 19) + 1,
        X \= 0,bats(Y), X \= Y.

get_room:- get_room.

stench:-
        you(X),wumpus(Z),connected(X,Y),connected(Y
,Z),!,X \= Z,
        write($You smell something funky and it's
one room away...$),nl.
stench:-
        you(W),wumpus(Z),connected(W,X),connected(X
,Y),
        connected(Y,Z),!,W \= Y,X \= Z,W \= Z,
        write($You smell something funky and it's
two rooms away...$),nl.
stench.

sound:-
        you(X),bats(Y),connected(X,Y),
        write($You hear a strange sound, like
flapping leather...$),nl.
sound.

draft:-
        you(X),pit(Y),connected(X,Y),!,
        write($A strange draft comes from
somewhere...$),nl.
draft.

arrow:-
        arrow(X),wumpus(X),retract(arrow(X)),
        write($You killed the WUMPUS! You
win!$),nl,end.
arrow:-
        arrow(X),pit(X),retract(arrow(X)),
        write($An echoing sound comes from room
#$),write(X),nl.
arrow:-
        arrow(X),bats(X),retract(arrow(X)),
        write($A squeeking sound comes from room
#$),write(X),nl.
arrow.

noarrows:-  arrows(0),write($You have no arrows! You
lose.$),nl,end.
noarrows.

go:- eaten,fall,carried,arrow,noarrows,fail.

go:-
        write($-> $),read(C),
        call(C),movewumpus,go.
go:-
        write($Ready.$),nl,go.

helpme:-
        nl,nl,
        write($Here are the commands you can
use:$),nl,
        write($goto(Room). = You move to room
#Room$),nl,
        write($shoot(Room). = You shoot an arrow
into room #Room.$),nl,
        write($observe. = Observe the
surroundings.$),nl,
        write($arrows. = Tells you how many arrows
you have left.$),nl,
        write($rooms. = Tells you what rooms you can
go into.$),nl,
        write($helpme. = This help section.$),nl,
        write($quit. = Exit the game.$),nl,nl.

goto(X):-
        you(Y),connected(X,Y),write($You go into
room #$),write(X),nl,nl,
        retract(you(Y)),assert(you(X)),observe.
goto(X):- write($I can't get to room
#$),write(X),nl,nl.

shoot(X):-
        you(Y),connected(X,Y),
        arrows(N), N \= 0, N1 is N - 1,
        assert(arrow(X)),retract(arrows(N)),
        assert(arrows(N1)),go.
shoot(X):- write($I can't shoot an arrow into room
#$),write(X),nl,nl.

quit:- write($Do you wish to exit(yes/no)?$),
        read(X),X=yes,halt.
quit:- observe.

end:- write($Do you wish to exit(yes/no)?$),
        read(X),X=yes,halt.
end:-
        retract(you(X)),assert(you(2)),
        retract(arrows(Y)),assert(arrows(5)),
        retract(wumpus(Z)),assert(wumpus(17)),
        observe,go.

observe:- you(X),write($You are in room
#$),write(X),nl,
        carried,arrows,rooms,sound,stench,draft.
observe:- write($That's all.$),nl,nl.

rooms:-
        you(X),connected(X,Y),
        write($You can go to room
#$),write(Y),nl,fail.
rooms.

arrows:-
        arrows(X),write($You have $),
        write(X),write($ arrows left.$),nl.

movewumpus:-
        wumpus(X),!,retract(wumpus(X)),
        connected(X,Y),assert(wumpus(Y)),
        write('Wumpus is now in '), write(Y).
movewumpus:- movewumpus.
```