Kuličky

logická hra naprogramovaná v ProLogu

Univerzita Hradec Králové
Fakulta informatiky a managementu
Informační management
Logické programování II

Petr Voborník

vobornik@mikmik.cz

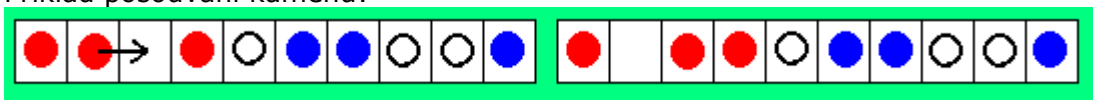
www.mikmik.cz

Obsah

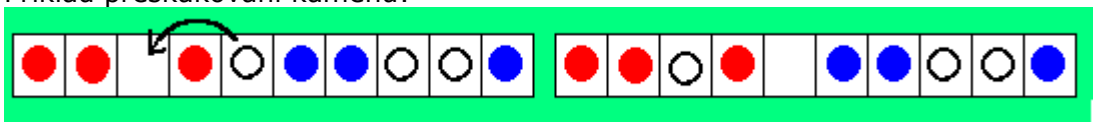
Zadání

Ke hře se používá zásobník s deseti místy, který obsahuje 9 kamenů. Tři kameny jsou bílé, tři modré a tři červené. Na začátku hry jsou kameny promíchané. Jeden hráč má jako své bílé kameny, druhý modré. Červené nepatří nikomu. Úkolem hráčů je dostat své kameny na svou stranu: bílé nalevo (1., 2., 3. pole), modré napravo (8., 9., 10. pole). Hráči se střídají v tazích. V každém tahu smí hráč pohnout svým nebo červeným kamenem. Pokud je sousední pole kamene prázdné, smí být kámen posunut do mezery. Pokud je mezi kamenem a mezerou jeden kámen, smí být tento kámen přeskočen. Vyhraje ten, kdo první dostane své kameny na svou stranu.

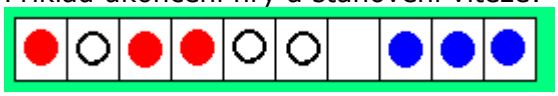
Příklad posouvání kamenů:



Příklad přeskakování kamenů:



Příklad ukončení hry a stanovení vítěze:



... vyhrál hráč s modrými kameny

Uživatelská dokumentace

Setup

Uživatel má možnost základního nastavení pro hru. Menu pro nastavení se spouští příkazem „**setup.**“. Poté se mu vypíše následující nabídka:

```
Soucasne nastaveni hry:
 Znaceni:
  b - bila
  m - modra
  c - cervena
  x - mezera
 Zacina: ten kdo naposled prohral
 b - hrac
 m - pocitac
 Pocitac promysli dopredu tahu 7,
 jinak je jeho nahodny faktor 50%.

Nastaveni muzete zmenit temito prikazy:
 Znaceni:
  sb - nastavit znak bile
  sm - nastavit znak modre
  sc - nastavit znak cervene
  sx - nastavit znak mezery
 Kdo zacne:
  zb - zacina bily
  zm - zacina modry
  zp - zacina ten kdo naposled prohral
  zv - zacina ten kdo naposled vyhral
 Kdo hraje:
  bp - bily bude pocitac
  bh - bily bude hrac
  mp - modry bude pocitac
  mh - modry bude hrac
 PC inteligence:
  pt - nastavit pocet dopredu promyslenych tahu
  pf - nastavit nahodny faktor
 k - ukoncit nastaveni
```

V první části je vypsáno současné nastavení a druhá část je seznam příkazů, které je možné použít, pro změnu nastavení hry.

„**Znaceni**“ říká, jak je která barva kuličky ve hře značena. „**Zacina**“ („**Kdo zacne**“) určuje, kdo bude ve hře začínat. „**Kdo hraje**“ informuje o tom, kdo bude za danou barvu hrát (počítač nebo hráč – člověk). V oddílu „**PC inteligence**“ znamená **počet tahu, které počítač promýšlí dopředu**, kolik tahu dopředu bude počítač promýšlet. Nedoporučuje se dávat více než 7, neboť na větší počet již většinou nestačí paměť. **Náhodný faktor** určuje, jakým podílem bude počítač vybírat tah náhodně, nenajde-li cestu vedoucí k jeho vítězství. Pokud je tento faktor příliš malý, může se stát, že hra uvízne na mrtvém bodě.

Nastavení hry ukončíte příkazem „**k.**“.

Hra

Samotnou hru spustíte příkazem „**hra.**“ Pokud jsou oba hráči počítač, tak před vašima očima proběhne rychlá hra, z níž vzejde vítěz. Je-li alespoň jeden z hráčů člověk, pak se vám vypíše zhruba takovéto zahájení hry:

```
Vitejte ve hre Kulicky.
Hra konci vytezstvim jednoho z hracu nebo zadanim znaku "k".
Bily hrac vyhrava pri umisteni vsech svych kulicek na levou
stranu,
modry pokud umisti vsechny sve kulicky do prava.
Hraci se stridaji a mohou mezerou "x" na misto svych nebo
cervenych kamenu.
Povolene tahy jsou: "l"  - pohyb mezery o jednu pozici do leva
                    "ll" - pohyb mezery o dve pozice do leva
                    "p"  - pohyb mezery o jednu pozici do prava
                    "pp" - pohyb mezery o dve pozice do prava
Preji peknou hru a hodne stesti!


Hraje hrac barvy b
Soucasny stav: [b,b,c,m,m,c,x,b,c,m]
Zadej smer, kterym chces tahnout [l,p,pp]:
```

Samotná hra pak probíhá tak, že se vypíše, který hráč je právě na tahu (znak jeho barvy), současný stav hracího pole a nabídka tahů, které hráč může vykonat (v hranaté závorce). Tyto tahy jsou odvozeny od současného stavu hracího pole a vyjadřují, jakým směrem a o kolik chcete táhnout (posunout) **mezerou** „x“. Pro usnadnění hry se totiž neposouvají kuličky, ale přesouvá se mezera, která je jen jedna a tudíž je snazší se ve hře orientovat. Možné posuny jsou **ll, l, p, pp**, přičemž je možné zadat jen ty, které neodporují pravidlům vzhledem k současnému stavu hracího pole. Každý hráč může mezeru prohodit pouze se svou nebo červenou kuličkou, vzdálenou maximálně dvě pozice.

„**l.**“ přesune mezeru s kuličkou o jednu pozici doleva, „**ll.**“ přesune mezeru s kuličkou o dvě pozici doleva, „**p.**“ přesune mezeru s kuličkou o jednu pozici doprava a „**pp.**“ přesune mezeru s kuličkou o dvě pozici doprava.

Pokud hráč nemůže táhnout žádným z těchto tahů, je o tomto informován a tah je opět předán hráči druhému.

Bílí hráč vítězí, obsadí-li svými kuličkami první tři levé pozice, modrý hráč vítězí, obsadí-li poslední tři pravé pozice. Hru je také možné předčasně ukončit zadáním „**k.**“.

Programátorská dokumentace

Součásti zdrojového kódu jsou četné komentáře a dobře volené názvy predikátů, které usnadňují jeho pochopení. Proto se zde budu zabývat pouze politikou jednotlivých částí programu, a funkcí jednotlivých predikátů jen okrajově.

Defaultní nastavení

Zde uvedené predikáty určují defaultní (výchozí) nastavení programu. Určují znaky jednotlivých barev a jejich počet v poli (ten není možné měnit, aby by hra i nadále fungovala správně). Dále definují další možnosti nastavení - setupu.

Setup

Tato část poskytuje uživateli možnost změnit některé parametry defaultního nastavení. Predikát **setup** vypíše současný stav a nabídku příkazů, počká na uživatelovu volbu a přes predikát **obsluz**, tento příkaz vykoná. Zadané hodnoty se ukládají do databáze. Pro smazání hodnot předchozích je využit příkaz *abolish*, který oproti *retract*u maže i predikáty zadané přímo ve zdrojovém kódu.

Hra

Nejprve je nutné vygenerovat hrací pole s dodržením určitých podmínek. Hrací pole o příslušném počtu polí musí obsahovat správné počty kuliček pro všechny tři barvy (což zajišťuje predikát **generuj**) a jeho výchozí stav nesmí být vítězným pro žádného z hráčů (to je zajištěno *repeat*em přímo ve spouštěcím predikátu **hra**).

Predikát **tahni_hrac** pak obsluhuje samotnou hru. Vypíše kdo je na tahu, současný stav hracího pole a vybídne hráče k tahu. Po jeho korektním zadání, předá tah druhému hráči zavoláním sama sebe s barvou protihráče.

Je-li na tahu počítač, je to rozpoznáno v predikátu **go_if_you_can** a program vybere a vykoná optimální tah sám.

Samotná hra se tedy spouští predikátem **hra** a je ukončena vítězstvím jednoho z hráčů, nebo zadáním „**k.**“.

PC inteligence

Inteligence počítače je řešena na třech úrovních. Prohledáváním do šířky (hledání vítězství), výběr podle seřazeného seznamu nejlepších tahů a náhodný výběr z možných tahů. Tyto tři možnosti zastřešuje predikát **pc_tah**, který nejprve zkusí prohledávání do šířky, nenajde-li tah vedoucí k jeho vítězství, pak náhodnou volbou rozhodne (náhodný faktor – toto procentuální vyjádření určuje, na kolik procent se přikloní ke třetímu způsobu), použije-li druhý nebo třetí způsob pro výběr svého tahu.

Prohledávání do šířky

Prohledávání do šířky vychází z aktuálního stavu hracího pole zadaného predikátu **hledej**. Najdou se všechny možné tahy z této pozice pro právě hrajícího hráče (nejprve počítače, poté jeho protihráče) a ty se všechny vykonají. Zkontroluje se, není-li některá pozice pro počítač vítězná a pokud ne, provede se to samé pro všechny tyto získané stavy hracího

pole. Jednotlivé cesty tahů se ukládají do fronty, která je stále navyšována, až do nalezení vítězného stavu, nebo vyčerpání hloubky rekurze.

Pokud by nějaká série tahů vedla k vítězství protihráče, je prohledávání této cesty ukončeno. Prohledává se podle pořadí předpokládaných nejlepších tahů (viz. úroveň PC inteligence).

Ke stavu hracího pole je na první pozici umístěn znak barvy toho hráče, který daný tah vykonal.

Výběr tahů ze seřazeného seznamu

Pro každého hráče je v predikátu **nejlepsi_tahy** definován sestupně seřazený seznam nejlepších tahů (pro bílého [pp,p,l,ll], pro modrého [ll,l,p,pp]). Poté je z možných tahů v predikátu **best_tah** vybrán tah, který je z těchto nejblíže počátku.

Tato strategie vychází z toho, že bílí hráč chce většinou doprava a modrý doleva. Není však bráno v úvahu možnost tahu s červenými kuličkami, své na své straně hráč nechce.

Při používání pouze tohoto způsobu výběru tahu, může dojít k „uvíznutí hry na mrtvém bodě“, kdy např. počítač táhne stále doleva a vy třeba nemáte jinou možnost, než jeho tah vrátit.

Náhodný výběr

Při náhodném výběru je predikátem **nahodny_tah** náhodně vybrán jeden ze všech možných tahů.

Závěr

Hru se mi podařilo naprogramovat bez větších potíží během zhruba dvou dnů (několika hodin čistého času). Nejdůležitějším faktorem zde je pravděpodobně inteligence počítače, o níž myslím, že je v rámci možností celkem dobrá.

Zdrojový kód programu

```
% ----------- defaultni nastaveni -----------

poli(10).
modra(m,3).
cervena(c,3).
bila(b,3).
nic(x,1).

max_scan(7).
nah_faktor(50).
pocitac(M):-modra(M,_).
zacina(zp).
vitez(M):-modra(M,_).

pocitac(xx). % aby byl tento predikat vzdy znam


% ----------- Setup -----------

kdo_zacina(zp):-write('ten kdo naposled prohral'),!.
kdo_zacina(zv):-write('ten kdo naposled vyhral'),!.
kdo_zacina(C):- write(C).

druh_hrace(B):-write(' '),write(B),write(' - '),fail.
druh_hrace(B):-pocitac(B),!,
               write('pocitac'),nl.
druh_hrace(B):-write('hrac'),nl.

% vypise soucasne nastaveni
vypis_stav:-write('Soucasne nastaveni hry:'),nl,
            bila(B,_),modra(M,_),cervena(C,_),nic(X,_),
            write(' Znaceni: '),nl,
            write('  '),write(B),write(' - bila'),nl,
            write('  '),write(M),write(' - modra'),nl,
            write('  '),write(C),write(' - cervena'),nl,
            write('  '),write(X),write(' - mezera'),nl,
            write(' Zacina: '),
            zacina(Z),kdo_zacina(Z),nl,
            druh_hrace(B),
            druh_hrace(M),
            write(' Pocitac promysli dopredu tahu '),
            max_scan(P),write(P),write(','),nl,
            write(' jinak je jeho nahodny faktor '),
            nah_faktor(N),write(N),write('%.'),nl.

vypis_menu:-write('Nastaveni muzete zmenit temito
prikazy:'),nl,
            write(' Znaceni:'),nl,
```

```
            write('  sb - nastavit znak bile'),nl,
            write('  sm - nastavit znak modre'),nl,
            write('  sc - nastavit znak cervene'),nl,
            write('  sx - nastavit znak mezery'),nl,
            write(' Kdo zacne:'),nl,
            write('  zb - zacina bily'),nl,
            write('  zm - zacina modry'),nl,
            write('  zp - zacina ten kdo naposled
prohral'),nl,
            write('  zv - zacina ten kdo naposled vyhral'),nl,
            write(' Kdo hraje:'),nl,
            write('  bp - bily bude pocitac'),nl,
            write('  bh - bily bude hrac'),nl,
            write('  mp - modry bude pocitac'),nl,
            write('  mh - modry bude hrac'),nl,
            write(' PC inteligence:'),nl,
            write('  pt - nastavit pocet dopredu promyslenych
tahu'),nl,
            write('  pf - nastavit nahodny faktor'),nl,
            write(' k - ukoncit nastaveni'),nl.

znak_is_ok(k):-write('"k" je rezervovany znak.'),nl,
               !,fail.
znak_is_ok(Z):-name(Z,S),
               delka(S,1),!.
znak_is_ok(Z):-write('Byl zadan neplatny znak: '),
               write(Z),nl,
               !,fail.

zadej_znak(Z,P):-repeat,
                  write('Zadej novy znak pro '),
                  write(P),write(':'),nl,
                  read(Z),
                 znak_is_ok(Z).


obsluz(k):-!.
obsluz(sb):-bila(_,P),
            zadej_znak(B,bilou),
            abolish(bila/2),
            asserta((bila(B,P))),!.
obsluz(sm):-modra(_,P),
            zadej_znak(M,modrou),
            abolish(modra/2),
            asserta((modra(M,P))),!.
obsluz(sc):-cervena(_,P),
            zadej_znak(C,cervenou),
            abolish(cervena/2),
            asserta((cervena(C,P))),!.
obsluz(sx):-bila(_,P),
            zadej_znak(X,mezeru),
```

```
            abolish(nic/2),
            asserta((nic(X,P))),!.
obsluz(zb):-abolish(zacina/1),
            asserta((zacina(B):-bila(B,_))),!.
obsluz(zm):-abolish(zacina/1),
            asserta((zacina(M):-modra(M,_))),!.
obsluz(zp):-abolish(zacina/1),
            asserta(zacina(zp)),!.
obsluz(zv):-abolish(zacina/1),
            asserta(zacina(zv)),!.
obsluz(bp):-modra(Mi,_),
            pocitac(Mi),!,
            abolish(pocitac/1),
            asserta((pocitac(B):-bila(B,_))),
            asserta((pocitac(M):-modra(M,_))),
            asserta(pocitac(xx)).
obsluz(bp):-abolish(pocitac/1),
            asserta((pocitac(B):-bila(B,_))),
            asserta(pocitac(xx)),!.
obsluz(bh):-modra(Mi,_),
            pocitac(Mi),!,
            abolish(pocitac/1),
            asserta((pocitac(M):-modra(M,_))),
            asserta(pocitac(xx)).
obsluz(bh):-abolish(pocitac/1),
            asserta(pocitac(xx)),!.
obsluz(mp):-bila(Bi,_),
            pocitac(Bi),!,
            abolish(pocitac/1),
            asserta((pocitac(B):-bila(B,_))),
            asserta((pocitac(M):-modra(M,_))),
            asserta(pocitac(xx)).
obsluz(mp):-abolish(pocitac/1),
            asserta((pocitac(M):-modra(M,_))),
            asserta(pocitac(xx)),!.
obsluz(mh):-bila(Bi,_),
            pocitac(Bi),!,
            abolish(pocitac/1),
            asserta((pocitac(B):-bila(B,_))),
            asserta(pocitac(xx)).
obsluz(mh):-abolish(pocitac/1),
            asserta(pocitac(xx)),!.
obsluz(pt):-zadej_tahy(X),
            abolish(max_scan/1),
            asserta(max_scan(X)),!.
obsluz(pf):-zadej_faktor(X),
            abolish(nah_faktor/1),
            asserta(nah_faktor(X)),!.
obsluz(X):-write('Byl zadan neznamy prikaz: '),
           write(X),getb(_),nl.
```

```
v_mezich(Min,Max,Co):-Co >= Min,
                      Co =< Max,!.
v_mezich(_,_,Co):-write('Zadana hodnota neni platna!'),nl,
                  !,fail.

zadej_tahy(X):-repeat,
                write('Zadej pocet tahu, ktere si pocitac bude
promyslet dopredu (0-10): '),nl,
                read(X),
               v_mezich(0,10,X).

zadej_faktor(X):-repeat,
                  write('Zadej nahodny faktor pro tahy
pocitace (0-100): '),nl,
                  read(X),
                 v_mezich(0,100,X).


setup:-repeat,
        vypis_stav,nl,
        vypis_menu,nl,
        write('Zadej prikaz: '),nl,
        read(X),
        obsluz(X),nl,
       X == k.


uloz_vyteze(B):-bila(B,_),!,
                abolish(vitez/1),
                asserta((vitez(B):-bila(B,_))).
uloz_vyteze(B):-abolish(vitez/1),
                asserta((vitez(M):-modra(M,_))).

ktery_zacina(Z):-zacina(zp),!,
                 vitez(V),
                 druhy_hrac(V,Z).
ktery_zacina(Z):-zacina(zv),!,
                 vitez(Z).
ktery_zacina(Z):-zacina(Z).


% ----------- Hra -----------


% pocet prvku X v seznamu S
pocet(_,[],0).
pocet(X,[X|T],P):-!,pocet(X,T,P0),
                  P is P0 + 1.
pocet(X,[_|T],P):-pocet(X,T,P).
```

```
% z nahodne vybraneho cisla X vrati prislusny objekt O
% a jeho maximalni povoleny pocet P v seznamu
indexy(X,O,P):-X < 3,modra(O,P),!.
indexy(X,O,P):-X < 6,cervena(O,P),!.
indexy(X,O,P):-X < 9,bila(O,P),!.
indexy(X,O,P):-nic(O,P).

% vybere nahodny objekt X a zjisti pocet P, kolikrat maximalne
% muze byt v seznamu
kdokoli(X,P):-C is rand(10),
              indexy(C,X,P).
kdokoli(X,P):-kdokoli(X,P).

% vymysli obkjekt X, ktery v seznamu S jeste neni vicekrat
% nez je mozne
vymysli_objekt(X,S):-kdokoli(X,M),
                     pocet(X,S,P),
                     P < M.

% generuj seznam delky D s patricnymi pravidly
generuj(0,[]).
generuj(D,[X|S]):-D1 is D - 1,
                  generuj(D1,S),
                  vymysli_objekt(X,S),!.

% polozka X ze seznamu S s indexem I
polozka(X,[X|_],0).
polozka(X,[_|T],I):-I1 is I - 1,
                    polozka(X,T,I1).

% nahrad polozku s indexem I v seznamu S hodnotou X do
% seznamu V
nahrad([],_,_,[]).
nahrad([_|T],X,0,[X|V]):-nahrad(T,X,-1,V),!.
nahrad([H|T],X,I,[H|V]):-I1 is I - 1,
                         nahrad(T,X,I1,V).

% prohodi prvky s indexy X, Y v seznamu S do vysledku V
prohod(S,X,Y,V):-polozka(Xp,S,X),
                 polozka(Yp,S,Y),
                 nahrad(S,Yp,X,V1),
                 nahrad(V1,Xp,Y,V).



% index I prvniho prveku hodnoty X v seznamu S
prvni(X,[X|_],0).
prvni(X,[_|T],I):-prvni(X,T,I1),
                  I is I1 + 1,!.
```

```
posun_o_kolik(l, -1,N):-N>0.
posun_o_kolik(p,  1,N):-poli(P),N<(P-1).
posun_o_kolik(ll,-2,N):-N>1.
posun_o_kolik(pp, 2,N):-poli(P),N<(P-2).

posun_na_znak(P,Z):-posun_o_kolik(Z,P,2).

% tahni posun smerem K (hraje hrac barvy B), seznam S, novy V
tahni(S,B,K,V):-nic(X,_),
                prvni(X,S,N),
                posun_o_kolik(K,P,N),
                N1 is N + P,
                n_ty(Bp,S,N1),
                pristupna_barva(B,Bp),
                prohod(S,N,N1,V).

% vyzada po uzivateli, aby zadal platny tah (barvu a smer)
% B barva hrace, Bt barva, kterou chce tahnout, St smer
%zadej_tah(B

% vrati yes, pokud zvolenou barvou muze hrac tahnout
pristupna_barva(B,B):-!.
pristupna_barva(_,X):-cervena(X,_),!.

% uskutecni zadany tah, pripadne napise, ze neni mozny
tahni_coment(S,B,K,T,V):-prvek(K,T),tahni(S,B,K,V),!.
tahni_coment(_,_,k,_,[]):-write('Hra byla ukoncena.'),nl,!.
tahni_coment(_,_,K,_,_):-write('Byl zadan neplatny tah "'),
                         write(K),write('"!'),nl,
                         !,fail.

% vrati barvu protihrace H do D
druhy_hrac(H,D):-modra(H,_),bila(D,_),!.
druhy_hrac(_,D):-modra(D,_).


% vyhral uz nekdo (M)?
vitez([B,B,B|_],B):-bila(B,_).
vitez(S,M):-modra(M,_),
            poli(P),P1 is P - 1,
            n_ty(M,S,P1),P2 is P1 - 1,
            n_ty(M,S,P2),P3 is P2 - 1,
            n_ty(M,S,P3).

% napise pokud nekdo vyhral a vrati yes
vyhra([]):-!.
vyhra(S):-vitez(S,M),write(S),nl,
          write('Vyhral '),write(M),write('!'),nl,
          uloz_vyteze(M).

vypis_barvu(B):-nl,write('Hraje hrac barvy '),write(B),nl.
```

```
vypis_stav(S):-write('Soucasny stav: '),write(S),nl.


% tah hrace nebo PC barvy B ze stavu S moznych tahu T do
vysledneho stavu V
go_if_you_can(S,_,[],S):-write('Hrac nemuze nijak
tahnout.'),nl,!.
go_if_you_can(S,B,T,V):-pocitac(B),!,
                        vypis_stav(S),
                        pc_tah(B,S,T,Kt),
                        tahni(S,B,Kt,V),
                        write('Pocitacuv tah: '),
                        write(Kt),write('.'),nl.
go_if_you_can(S,B,T,V):-repeat,
                         vypis_stav(S),
                         write('Zadej smer, kterym chces
tahnout '),
                         write(T),write(':'),nl,
                         read(Kt),
                        tahni_coment(S,B,Kt,T,V).

% vstupni pole je seznam S, tahne hrac (zivi) barvy B
tahni_hrac(S,B):-vypis_barvu(B),
                 tahy(S,B,T),
                 go_if_you_can(S,B,T,V),!,
                 druhy_hrac(B,Dh),
                 (vyhra(V);tahni_hrac(V,Dh)).

mensi(A,B,A):-A<B,!.
mensi(A,B,B).
vetsi(A,B,A):-A>B,!.
vetsi(A,B,B).

% seznam moznych tahu v poli S, hrace Hr do vysledku V
tahy(S,Hr,V):-nic(X,_),
              prvni(X,S,N),
              Ni is N - 2,
              vetsi(Ni,0,Ni1),
              poli(P),P1 is P - 1,
              Na is N + 2,
              mensi(Na,P1,Na1),
              tahy(S,Hr,Ni1,Na1,0,V).
tahy(_,_,_,Na,N,[]):-N>Na.
tahy([_|T],Hr,Ni,Na,N,V):-N<Ni,
                          N1 is N + 1,
                          tahy(T,Hr,Ni,Na,N1,V).
tahy([H|T],Hr,Ni,Na,N,[K|V]):-N>=Ni,N=<Na,
                              pristupna_barva(Hr,H),!,
                              posun_z_pozice(Ni,N,Na,P),
                              posun_na_znak(P,K),
                              N1 is N + 1,
```

```
                              tahy(T,Hr,Ni,Na,N1,V).
tahy([_|T],Hr,Ni,Na,N,V):-N>=Ni,N=<Na,
                          N1 is N + 1,
                          tahy(T,Hr,Ni,Na,N1,V).

% minimim Ni, maximum Na, pozice N => posun P
posun_z_pozice(Ni,N,Na,P):-Ni = 0,Na >= 4,
                           P is N - 2.
posun_z_pozice(Ni,N,Na,P):-Ni = 0,Na < 4,
                           P is N - Na + 2.
posun_z_pozice(Ni,N,Na,P):-Ni > 0,
                           P is N - Ni - 2.



hra:-write('Vitejte ve hre Kulicky.'),nl,
     write('Hra konci vytezstvim jednoho z hracu nebo zadanim
znaku "k".'),nl,
     write('Bily hrac vyhrava pri umisteni vsech svych kulicek
na levou stranu,'),nl,
     write('modry pokud umisti vsechny sve kulicky do
prava.'),nl,
     write('Hraci se stridaji a mohou mezerou "x" na misto
svych nebo cervenych kamenu.'),nl,
     write('Povolene tahy jsou: "l"  - pohyb mezery o jednu
pozici do leva'),nl,
     write('                    "ll" - pohyb mezery o dve
pozice do leva'),nl,
     write('                    "p"  - pohyb mezery o jednu
pozici do prava'),nl,
     write('                    "pp" - pohyb mezery o dve
pozice do prava'),nl,
     write('Preji peknou hru a hodne stesti!'),nl,nl,
     poli(P),
     bila(B,_),
     modra(M,_),
     ktery_zacina(C),
     repeat,
      generuj(P,S),
     not(vitez(S,B)),
     not(vitez(S,M)),
     tahni_hrac(S,C).


% ----------- PC inteligence -----------


% setupne serazeny seznam nejlepsich tahu pro barvu B
%nelepsi_tahy(B,S).
nejlepsi_tahy(B,[pp,p,l,ll]):-bila(B,_),!.
nejlepsi_tahy(_,[ll,l,p,pp]).
```

```
% prvni mozny V tah z mnozny M podle doporucenych D
%prvni_mozny(M,D,V).
prvni_mozny(M,[H|_],H):-prvek(H,M),!.
prvni_mozny(M,[_|T],V):-prvni_mozny(M,T,V).


% nejlepsi tah V pro barvu B, z moznych tahu T
best_tah(_,[H|[]],H).
best_tah(B,T,V):-nejlepsi_tahy(B,S),
                 prvni_mozny(T,S,V).



% seradi tahy do V ze S podle rady R
% serad_tahy(R,S,V).
serad_tahy([],_,[]).
serad_tahy([H|T],S,[H|V]):-prvek(H,S),!,
                           serad_tahy(T,S,V).
serad_tahy([H|T],S,V):-serad_tahy(T,S,V).


% najde vsechny mozne pozice po danem tahu
pristi_tah(_,_,_,_,[],[]).
pristi_tah(X,P,Bs,Bt,[H|T],[Qv|V]):-tahni(X,Bt,H,W),
                                     Q = [Bt|W],
                                     not(prvek(Q,P)),
                                     not(vitez(W,Bs)),!,
                                     pristi_tah(X,P,Bs,Bt,T,V),
                                     spoj([Q],P,Qv).
pristi_tah(X,P,Bs,Bt,[H|T],V):-tahni(X,Bt,H,W),
                               pristi_tah(X,P,Bs,Bt,T,V).


po_tahu(P,[],P):-!.
po_tahu(_,S,S).


% hledej do sirky cestu pro vyhru hrace barvy B
% B - pro koho se hleda vyhra, Bt - prave tahne,
% S - seznam stavu, O - statni cesty
hledej(_,S,_,_):-delka(S,D),
                 max_scan(M),
                 M >= D,!,fail.      % proti preteceni
hledej(B,[H|T],O,[H|T]):-H = [_|X],
                         vitez(X,B),!.
hledej(B,[P|J],O,W):-P = [Bt|X],
                     druhy_hrac(B,Bs),   % Bs souper
                     druhy_hrac(Bt,Bn),  % Bn hraje now
                     tahy(X,Bn,M),
```

```
                    nejlepsi_tahy(Bn,M1),
                    serad_tahy(M1,M,M2),
                    pristi_tah(X,[P|J],Bs,Bn,M2,Vd),
                    spoj([[Bn|X]],[P|J],Vn),
                    po_tahu([Vn],Vd,V),
                    spoj(O,V,F),
                    F = [H|T],
                    hledej(B,H,T,W).


% o kolik V se pohlo "x" v D oproti S
pohyb_x(S,D,V):-nic(X,_),
                prvni(X,S,Is),
                prvni(X,D,Id),
                V is Id - Is.

% prohleda do sirky a nejde-li cestu k vitezstvi, vrati jeji
prvni tah
tah_hledanim(B,S,V):-druhy_hrac(B,Bt),
                    spoj([Bt],S,Sh),
                    hledej(B,[Sh],[],C),
                    predposledni(D,C),
                    pohyb_x(Sh,D,P),
                    posun_na_znak(P,V).


% zvoli nahodny tah V z T
nahodny_tah([V],V):-!.
nahodny_tah(T,V):-delka(T,D),
                  I is int(rand(D)),
                  n_ty(V,T,I).


% najde nejvhodnejsi tah pro PC pomoci jedne z metod
pc_tah(B,S,_,V):-tah_hledanim(B,S,V),!.
pc_tah(B,_,T,V):-R is rand(100),
                 nah_faktor(N),
                 R > N,
                 best_tah(B,T,V).
pc_tah(_,_,T,V):-nahodny_tah(T,V).



% ----------- Pomocne predikaty -----------

prvek(X,[X|_]).
prvek(X,[_|T]):-prvek(X,T).

n_ty(X,[X|_],0).
n_ty(X,[_|T],N):-N1 is N - 1,
```

```
                n_ty(X,T,N1).

spoj([],B,B).
spoj([H|T],B,[H|V]):-spoj(T,B,V).

delka([],0).
delka([_|T],V):-delka(T,V1),
                V is V1 + 1.

predposledni(X,[X|[_]]).
predposledni(X,[_|T]):-predposledni(X,T).
```