Prolog 2-1

opakování základů Prologu

%význam ... rodic(X,Y). ..... X=dite, Y=rodic

```
rodic(jan,dana).
rodic(jan,zdenek).
rodic(dana,jirina).
rodic(dana,antonin).
rodic(zdenek,marie).
rodic(zdenek,cestmir).
rodic(milos,jirina).
rodic(milos,antonin).
rodic(petr,jitka).
rodic(petr,milos).

muz(jan).
muz(zdenek).
muz(antonin).
muz(cestmir).
muz(milos).
muz(petr).

zena(dana).
zena(jirina).
zena(marie).
zena(jitka).
```

Pradodič:

```
prarodic(X,Y) :-
  rodic(X,Z),
  rodic(Z,Y).
```

příklad:

```
   prarodic(X,Y). => X=jan,Y=jirina
                     X=jan,Y=antonin
                     X=jan,Y=marie
                     X=jan,Y=cestmir
                     X=petr,Y=jirina
                     X=petr,Y=antonin
```

Babička:

```
babicka(X,Y) :-
  prarodic(X,Y),
  zena(Y).
```

příklad:

```
   babicka(X,Y). => X=jan,Y=jirina
                    X=jan,Y=marie
                    X=petr,Y=jirina
```

Prolog 2-2

Bratr:
```
bratr(X,Y) :-
  sourozenec(X,Y),
  muz(Y).
```

příklad:
```
   bratr(X,Y). => X=dana, Y=milos
```

Sestřenice:
```
sestrenice(X,Y) :-
  rodic(X,R),
  sourozenec(R,S),
  rodic(Y,S),
  zena(Y).
```

příklad:
```
   sestrenice(X,Y). => no
```

<u>příklady řešené pomocí přímé rekurze nebo použití append</u>

Přidá prvek na konec seznamu rekurzivně:
```
pridej(X,[],[X]).
pridej(X,[H|L],[H|NewL]) :-
  pridej(X,L,NewL).
```

příklad:
```
   pridej(a,[b,c,d],X). => X=[b,c,d,a]
```

Přidá prvek na konec seznamu pomocí append:
```
pridej_A(X,L,NL) :-
  append(L,[X],NL).
```

příklad:
```
   pridej_A(a,[b,c,d],X). => X=[b,c,d,a]
```

Vrátí poslední prvek seznamu:
```
last(L,X) :-
  pridej(X,_,L).
```

příklad:
```
   last([a,b,c],X). => X=c
```

Vrátí poslední prvek seznamu pomocí append:
```
last_A(L,X) :-
  append(_,[X],L).
```

příklad:
```
   last_A([a,b,c],X). => X=c
```

Prolog 2-3

Vrátí seznam bez posledního prvku:

```
butlast(L,S) :-
  pridej(_,S,L).
```

příklad:
   butlast([a,b,c],X). => X=[a,b]

Vrátí seznam bez posledního prvku pomocí append:

```
butlast_A(L,S) :-
  append(S,[_],L).
```

příklad:
   butlast_A([a,b,c],X). => X=[a,b]

Otočení seznamu:

```
reverse([],[]).
reverse([X|L],RevXL) :-
  reverse(L,RL),
  append(RL,[X],RevXL).
```

příklad:
   reverse([1,2,3,4],X). => X=[4,3,2,1]

Member pomocí append:

```
member_A(X,L) :-
  append(_,[X],K),
  append(K,_,L).
```

příklad:
   member_A(c,[a,b,c]). => yes

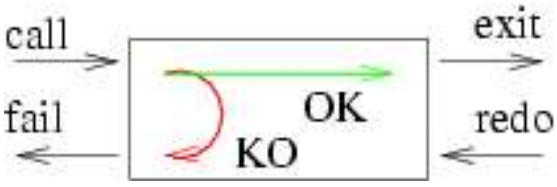
Smaž všechny výskyty daného prvku :

```
del_all(X,[],[]).
del_all(X,[X|L],NewL) :-
  del_all(X,L,NewL).
del_all(X,[Y|L],[Y|NewL]) :-
  ruzne(X,Y),
  del_all(X,L,NewL).
```

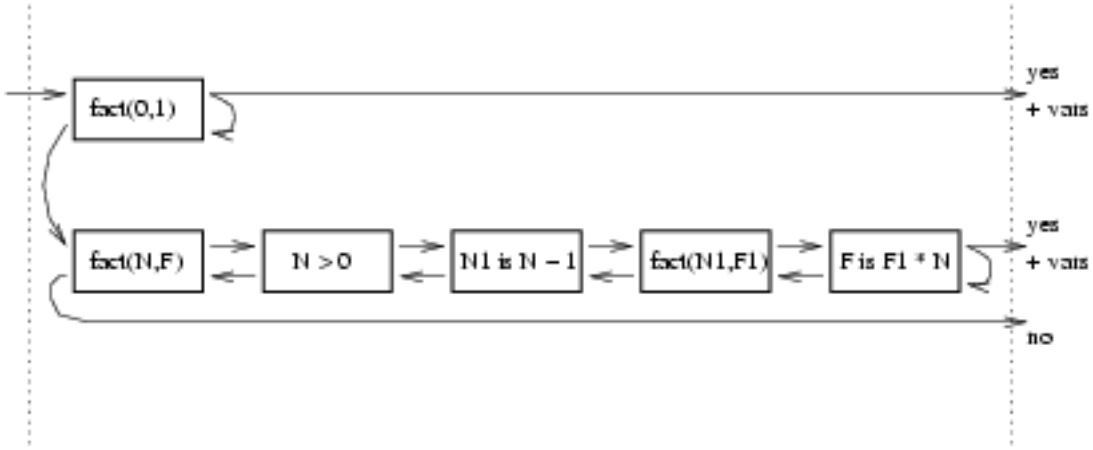
příklad:
   del_all(1,[1,2,1,3,1],X). => X=[2,3]

Prolog 2-4

Box model Prologu




```
fact(0,1).
fact(N,F) :-
        N>0,
        N1 is N - 1,
        fact(N1,F1),
        F is F1 * N.
```



Řezy

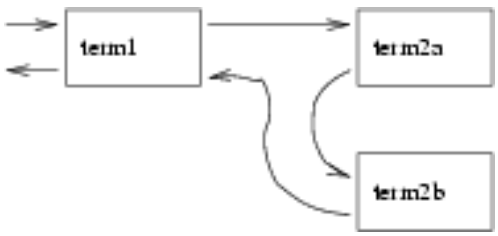
- řez přerušuje redo větev, zapisuje se pomocí vykřičníku !

zelený řez - program vrací stejné výsledky, jenom jsem zmenšil vyhledávací prostor a tím napsal efektivnější program

červený řez - ořezávám i část výsledků

Středník jako logická spojka OR

term1, (term2a ; term2b)



Výběr největšího prvku ze seznamu:

```
max1(X,Y,X) :-
  X >= Y.
max1(X,Y,Y) :-
  X < Y.
```

příklad:

```
   max1(2,5,X). => X=5
   max1(5,2,X). => X=5
```

Výběr nějvětšího prvku ze seznamu pomocí řezu:

```
max2(X,Y,X) :-
  X >= Y,
  !.
max2(X,Y,Y).
```

příklad:

```
   max2(2,5,X). => X=5
   max2(5,2,X). => X=5
```

Zploštění strukturovaného seznamu:

```
zplosti([],[]).
zplosti([H|T],L) :-
  zplosti(H,NewH),
  zplosti(T,NewT),
  append(NewH,NewT,L),
  !.
zplosti(X,[X]).
```

příklad:

```
   zplosti([[[a],b],[c,[d]]],X). => X=[a,b,c,d]
```

Množinové operace se seznamy

Test na podseznam:

```
podseznam(S,L) :-
  append(_,S,K),
  append(K,_,L).
```

příklad:

```
   podseznam([a,b],[a,b,c]). => yes
```

Průnik dvou seznamů:

```
prunik([],B,[]).
prunik([H|A],B,[H|AB]) :-
  member(H,B),
  prunik(A,B,AB),
  !.
prunik([H|A],B,AB) :-
  prunik(A,B,AB).
```

příklad:
prunik([a,b,c],[b,c,d],X). => X=[b,c]

Sjednocení dvou seznamů:

```
sjedn([],B,B).
sjedn([H|A],B,AB) :-
  member(H,B),
  sjedn(A,B,AB),
  !.
sjedn([H|A],B,[H|AB]) :-
  sjedn(A,B,AB).
```

příklad:
sjedn([a,b,c],[b,c,d],X). => X=[a,b,c,d]

Rozdíl dvou seznamů:

```
rozdil([],B,[]).
rozdil([H | A],B,AB) :-
  member(H,B),
  rozdil(A,B,AB),
  !.
rozdil([H | A],B,[H | AB]) :-
  rozdil(A,B,AB).
```

příklad:
rozdil([a,b,c],[b,c,d],X). => X=[a]

Prolog 2-7

Selhání
- v některých případech můžeme pro postup daným pravidlem požadovat jeho nesplnění .. pomocí fail.

Různé parametry:
```
ruzne(X,X) :-
  !,
  fail.
ruzne(X,Y).
```

příklad:
ruzne(5,5). => no
ruzne(1,2). => yes
ruzne(3,X). => no %unifikace se provede úspěšně, zavede nás do slepé uličky k fail (protože výsledek je NO, nedochází k výpisu nastavení proměnné)

- první pravidlo můžeme považovat jako slepou uličku do které se cíl dovede pokud jsou oba argumenty shodné (= unifikovatelné); za řezem následuje explicitní definitivní selhání.
- pokud argumenty shodné nejsou použije se druhá klauzule, která má vždy úspěch

Negace jako selhání:
```
not(X) :- call(X), !, fail.
not(X).
```

- cíl not(X) selže, pokud cíl X je prologovsky splnitelný