Softwarové technologie

[Logický program 2](#_Toc164079136)

[Predikáty 2](#_Toc164079137)

[Syntaxe Prologu 3](#_Toc164079138)

[Databáze 6](#_Toc164079139)

# Logický program

* Jazyk pro programování symbolických výpočtů
* Založen na predikátové logice
  + "x(má\_rád(x,pivo) Þmá\_rád(Honza, x))
  + Prolog: má\_rád(Honza, X) :- má\_rád(X, pivo)
* Název odvozen z PROgramování v LOGice
* Jeho úspěch pro vznik nové disciplíny matematické informatiky => logického programování
  + Specifické oblasti použití
  + Umělá inteligence
  + Znalostní inženýrství
  + Databázové a expertní systémy
  + Podpora specializovaných činností, např. projektování (CAD), výuce (CAI) aj.
* Porovnání s konvenčními jazyky
  + Jde o neprocedurální jazyk
  + Lze se více soustředit na popis vlastností relací – tedy co se má vypočítat
  + Bez nutnosti řešit, jak se to má udělat
  + Deklarativní
    - Při psaní programu deklarujeme fakta a pravidla pro popis vlastností a vztahů
  + Konverzační
    - Uživatel klade dotazy
    - Prolog odpovídá
  + Interaktivní
    - Pokud uživatel dotazy neklade, Prolog nepracuje a čeká
* V Prologu nejsou příkazy pro řízení běhu výpočtu ani příkazy pro řízení toku dat
* Chybějí prostředky pro programování cyklů, větvení, …
* Nepoužívá se přiřazovací příkaz
* Rozdílná úloha proměnných
  + Proměnná v Prologu označuje po dobu výpočtu objekt, který vyhovuje určitým podmínkám
  + Jeho vymezení se při výpočtu upřesňuje
* Průběh výpočtu v Prologu řízen jeho interpretem na základě znění programu
* Programátor může chod výpočtu ovlivnit řídícími příkazy v mnohem menší míře než u jiných jazyků
* Původně navržen jako specializovaný na symbolické výpočty, moderní implementace směřují k obecnějšímu použití

## Predikáty

* Vyjadřují fakta, která mají význam vztahů mezi objekty
* Píše se stejně jako funktor
* 2 základní režimy
  + Konzultační režim
    - Slouží k samotnému vytváření programu
  + Dotazovací režim
    - Kladou se otázky pomocí predikátů
  + Otázka vždy začíná otazníkem
  + Další výsledek se prolog pokusí najít po zadání středníku, stisknutím samotného enteru prolog ukončí dotazování

## Syntaxe Prologu

* Prolog se skládá z termů
* Jako termy se označují konstanty, proměnné, struktury
* Konstanty
  + Integer
    - Celá čísla
    - Rozsah záleží na implementaci
    - Většina moderních implementací i s reálnými
  + Atomy
    - Sekvence znaků začínající malým písmenem
    - Nebo řetězec uzavřený v apostrofech
    - Speciální atomy: ? – :- ! .
* Proměnné
  + Sekvence znaků začínající velkými písmeny nebo podtržítkem
  + Samotné podtržítko = anonymní proměnná
    - Její hodnota není podstatná
    - Nebude dále využívána
* Struktury
  + Objekt skládající se z jiných objektů
  + Komponenty lze spojovat pomocí funktoru
* Term je zapsán jako sekvence znaků
  + Malá a velká písmena
  + Číslice
  + Speciální znaky
* Vestavěné predikáty
  + Procedury nabízené danou implementací Prologu
  + Velké množství predikátů nejen pro výpis na obrazovku a také získávání dat od uživatele, ale také na klasifikaci a konverzi termů, groupování a ladící příkazy
  + Jsou považovány za termy
* Konstanty a proměnné se označují jako atomické termy
  + Z atomických termů se dají vytvářet složitější výrazy = struktury
* Programování v Prologu spočívá
  + V deklarování faktů o objektech a relacích mezi nimi
    - Fakta
      * Nepodmíněné výrazy, vyjadřují skutečnost o relaci
      * Jde o vztahy, které mají jméno, za kterým následuje v kulatých závorkách výčet objektů, kterých se vztah týká
      * Každý fakt je ukončen tečkou
      * matka(Matka, Dite).
  + Definování pravidel
    - Podmíněné výrazy
    - Dvě části, oddělené :-
    - V levé části je hlava pravidla, v pravé tělo pravidla
      * sourozenci(X,Y) :- matka(M,X), matka(M,Y)
  + Kladení a zodpovídání dotazů
    - Začíná dvojicí znaků ?-, končí tečkou
    - Dotaz, který neobsahuje proměnné se nazývá základní otázka
    - Výsledkem je odpověď
      * Výpis hodnot proměnných, které jsou součástí otázky yes/no
    - No může znamenat
      * Zápornou odpověď
      * Neexistující odpověď
* Seznamy
  + Prakticky jedinou předdefinovanou datovou strukturou
  + Tvořen posloupností prvků
    - Oddělených čárkami
    - A uzavřených do hranatých závorek
  + Vytváření
    - Funktor .
  + Seznam se skládá z hlavy a těla
    - Výjimkou je prázdný seznam
  + Syntaktická pomůcka
    - Svislítkem lze oddělit hlavu od těla
    - [ H | T ]
  + Operace se seznamy
    - Append – spojování
    - Member – zjistí, zda je prvek členem
    - Length – délka seznamu
    - Delete – odstraní prvek ze seznamu
  + Seznam je rekurzivně definovaná struktura
    - I procedury s ním pracující jsou rekurzivní
* Databáze
  + Můžeme s výhodou použít tam, kde potřebujeme aktivně přistupovat k bázi znalostí
  + Největší předností je možnost měnit program během jeho běhu
* Řízení databáze
  + Assert, asserta, assertz – přidá do DB
  + Retract, retractall – odstraní z DB
  + Listing – výpis klauzulí v DB
  + Clause – hledá klauzule
* Řízení průchodu programem
  + ! (řez)
    - Představuje klíčové rozhodnutí
    - Překročí-li jej, už se nevrací
    - Červený
      * Pokud jej z programu smažu, bude program pracovat chybně
    - Zelený
      * Pokud jej smažu, může dojít ke snížení efektivity, ale funkce se nezmění
  + Fail
    - Nikdy nesplněný predikát
    - Nelze se přes něj dostat
  + Repeat
    - Při průchodu programem blokuje návrat
    - Nekonečný cyklus
      * Repeat
      * Write(‘Pracuj’), nl,
      * Fail.

# Databáze