1. ОСНОВИ НА ПРОЛОГ

Въведение. Пролог е език за програмиране, който съществено се различава от останалите езици за програмиране. Пролог (PROgramming in LOGic) означава език за логическо програмиране. Първата официална версия на Пролог е развита от университета Марсейл във Франция от Алан Колмераул през 1970 г. като инструмент за логическо програмиране. Днес Visual Prolog е на пазара наравно с програмни системи като SQL Database Systems, C++ development systems и други езикови среди като Visual Basic, Borland's Delphi, or IBM's Visual Age. Версиите на компилатора се поддържат и развиват от Prolog Development Center с необходимата документация и приложени примерни програми, разработени от Prolog Community. Най-новата версия на средата Visual Prolog 7.1 Personal Edition for Windows Vista/XP/2000/NT, цялата документация и много примери можете да намерите и изтеглите безплатно на интернет адреса на Prolog Development Center www.visual-prolog.com. На същия адрес можете да поръчате и закупите Visual Prolog 7.1 Commercial Edition for Windows Vista/XP/2000/NT.

В основата на езика е формална система за описание на “неща” и връзките между тях, наречена логика на клаузите на Хорн. Под “неща” се разбира обект, лице, понятие или почти всичко, което в естествения език се изразява най-често чрез съществително име.

Ако в естествения език (ЕЕ) се казва “Иван има сестра Мима”, записано на Пролог това иглежда така: has_sister(“Иван”, “Мима”).

has_sister се нарича **предикат** (отношение) с два аргумента, от които на първа позиция е притежаващият, а на втора - сестрата. Такива конструкции се наричат **факти**. Конструкцията започва с отношението и след него в скоби се записват двете „неща”.

Отбележете, че редът в изреждането на нещата е много важен. Приет веднъж, този ред не трябва да се изменя.

В ЕЕ съществуват и по-сложни конструкции, например “Сашо има сестра X, ако Петър има сестра X”. Записана на Пролог, тази конструкция се нарича **правило** и изглежда по следния начин:

```
has_sister(“Сашо”, X):- has_sister(“Петър”, X).
```

Знакът “:-” замества връзката “ако” в правилото.

Нека множество от факти и правила наречем “свят”. Ето един пример за един малък свят.

```
has_sister(“Иван”, “Мима” ).
has_sister(“Краси”, “Ани” ).
has_sister(“Петър”, “Алекса” ).
has_sister(“Сашо”, X):- has_sister(“Петър”, X).
has_sister(“Георги”, X):- has_sister(“Сашо”, X).
```

В него има 9 означени с малките си имена лица , 3 факта и 2 правила. Променливите се означават с главни букви на латиница. Анонимната променлива (променлива без име) се означава с подчертаващо тире (_). Кавичките се използват за означаване на стринг.

Пролог ни предоставя възможност да изучаваме този свят, задавайки въпроси от вида:

1) Има ли Краси сестра Ани?
2) Има ли Краси сестра?
3) На кого е сестра Мима?
4) Коя е сестрата на Сашо?

Записани на Пролог, тези въпроси се наричат **цели** (Goal) и изглеждат по следния начин:

1) Goal:
has_sister(“Краси”,”Ани”)

В някои диалекти на езика записът е следният

?-has_sister(“Краси”,”Ани”)

2) Goal:
has_sister(“Краси”, _)
3) Goal:
has_sister(X,”Мима”)
4) Goal:
has_sister (“Сашо”, X)

Всяка клауза завършва с точка.

Когато се задава целта, не се поставя точка.

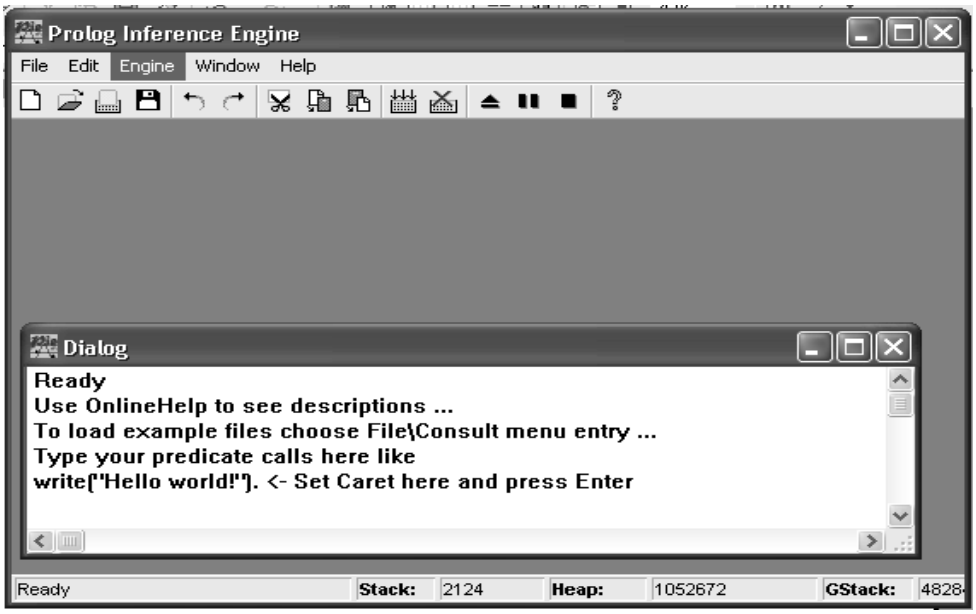
Фактите, правилата и целите изчерпват понятието клаузи на Хорн.

Една програма на Пролог се състои от описание на свят и поставена цел. При стартирането си програмата се стреми на намери целта в рамките на описания свят. Възможни са следните случаи:

а) отговорът на въпроса е “Yes” с едно или повече решения.

б) отговор на въпроса в рамките на описания свят няма и машината съобщава за този факт с “No”

Постановка. Ще използваме елементарен интерпретатор за проиграване на малки програми на Пролог (PIE – Prolog Inference Engine- Пролог машина за извод). След стартиране на PIE върху екрана се появява:



PIE работи с описание на света (факти и правила - клаузи), подредено в текстов файл.

За да създадете нов текстов файл, изберете от меню **File** командата **New**. В появилия се прозорец запишете клаузите на вашата програма на Пролог. След това укажете на PIE, че трябва да работи с клаузите от този текстов файл. Това се прави с команда **Consult** от менюто **Engine**. Машината за извод е готова да получава въпроси-цели в диалоговия прозорец, който се е показал по подразбиране.

Въпросът се записва на нов ред (под Goal:), след което се натиска клавишът **Enter.** За показаните по-горе въпроси върху екрана ще се изпише следният диалог:

Goal:
has_sister(“Краси”,”Ани”) →това е въпросът
Yes → това е отговорът на компютъра
1 solution → посочва се и броят на намерените решения

Goal:
has_sister(“Краси”, _)
Yes
1 solution

Goal:
has_sister(X,”Мима”)
X=”Иван”
1 solution

Goal:
has_sister (“Сашо”, X)
X=Алекса
1 solution

Нека проследим как Пролог машината работи, за да намери решението на конкретен въпрос.

Първият въпрос съдържа само имена и машината само обхожда клаузите и търси пълно съвпадение. Такова съвпадение е намерено и машината отговаря с *Yes*.

При втория въпрос на втора позиция машината намира анонимната величина “_”. Тя обхожда клаузите, като търси факт, който дава съвпадение без да се отчита втората позиция на фактите. При откриване на първото такова съвпадения машината генерира отговор Yes и спира обхождането

При третия въпрос различният момент е, че машината ще обходи всички факти и ще изведе толкова отговори като стойност на променливата *X*, колкото частични съвпадения (без отчитане на първа позиция в клаузта) е открила. Замяната на променливата *X* с конкретно име се нарича **унификация**.

Най-сложно е търсенето при четвъртия въпрос. Машината не намира търсеното съвпадение сред фактите, но открива правило, според което целта **has_sister (“Сашо”, X),** може да се замени с нова цел **has_sister(“Петър”, X)**. Машината поема новата цел и при новото търсене сред фактите унифицира променливата *X* с ”Алекса”. Новите цели в този процес се наричат **подцели**, а унифицирането на променливата с конкретна стойност я превръща от свободната променлива в свързана променлива.

В заключение, Пролог има вградени механизми за обхождане на клаузите, за откриване на пълни и частични съвпадения, за построяване на последователности от цели и подцели и за унифициране на променливи.

Всъщност Пролог е декларативен език и като такъв се основава на следните идеи:

- описва се задачата, но без да се посочва пътят за нейното решаване,
- програмата работи независимо от реда на подреждане на клаузите.

И така вие направихте своята първа програма на Пролог.

Можете да запишете вашата програма като текстов файл, за да я използвате и друг път – меню **File** , команда **New**.

Ако желаете да стартирате преди това запомнена в текстов файл програма, трябва да заредите съответния й файл в машината за извод **File -> Consult.** При необходимост да редактирате програмата тя трябва да се зареди в прозореца за редактиране чрез **Engine -> Reconsult**

За да проследите изпълнението на програмата по стъпки, от меню **Engine** изберете **Trace Calls** (Ctrl+t). Когато изберете **Trace Calls** повторно, проследяването ще се изключи.

Задание

1. Проследете изпълнението на програмата по стъпки при търсенето на отговора на горните въпроси.

2. Обогатете вашия свят с нови роднински връзки, например баща и дядо.

Забележка. Всяко правило се състои от глава и тяло. Главата се отделя от тялото със знака (:-), който се чете “ако”. Главата ще бъде истина, ако тялото е истина. Наред с простите правила (виж примерите в текста) могат да се записват и съставни правила. Например:

grandFather(X, GrandFather) :- father(X, Father), father(Father, GrandFather).

При съставните правила запетаята (,) означава логическо “И”, точка и запетая (;)- логическо “ИЛИ”, а not - логическо “НЕ”.

3. Конструирайте свят, в който конкретен въпрос има няколко отговора и трасирайте намирането на тези отговори.

Какво ще се измени в трасировката, ако поставите предиката cut (!) - отбелязва се с удивителна – на различни места сред клаузите на програмата?

Обосновете следните свързани с използването на cut(!) твърдения:

- редът на записване на клаузите не е без значение;

- програмата вече не е чисто декларативна;
- трябва да имате ясна представа за хода на изпълнението.

-ползата от този оператор е, че в редица случаи изпълнението на програмата значително се ускорява.

4. Направете отговорите на компютъра по-леки за възприемане чрез предиката write(“стринг за отпечатване...\n”).

Забележка. Предикатът write(“стринг за отпечатване...\n”) се използва за отпечатване на съобщения на екрана. Този предикат се изпълнява еднократно и също като предиката cut (!) спира по-нататъшното търсене на други решения.

5. Използвайте предиката fail, за да елиминирате вредните ефекти от включването в програмата на предиката cut (!).

Забележка. Изпълнението на предиката fail генерира стойност false и заставя програмата да продължи проследяването за търсене на друго решение до изчерпване на всички решения. Казва се, че той “подсилва” проследяването.

6. Разширете вашия свят с понятията майка (mother) и баба (grandmother) и постройте фамилно дърво.

Дефинирайте понятието родител (parent) по следния начин:

```
parent(X, Parent) :-
   mother(X, Parent);
   father(X, Parent).
```

Защо родителят на дадено лице се появява по два пъти сред отговорите?

Защо не се препоръчва използването на (;) в програмите?

Забележка. Запетаята (,) и точката и запетаята (;) си приличат и е за предпочитане единият знак да не се използва.

Вместо горния запис е за предпочитане неговият еквивалент:

```
parent(X, Parent) :-    mother(X, Parent).
parent(X, Parent) :-    father(X, Parent).
```

7. Опитайте се да построите отново същия свят, като добавите пола на лицата.

Пример:

```
person(“Иван”, "male").
person(“Мима”, "female").
```

Забележка. Колкото по-съществена е една връзка в описвания свят, толкова по-рано тя трябва да се появи в описанието.

8. Като имате предвид горната задача, съставете програма на Пролог, която открива телефонните номера по зададено име.

Задайте правило, при изпълнението на което да е необходимо първо да се намери телефонът на Иван и да се установи, че телефонът на Петър е същия.

Задайте правило, което гласи, че телефонът на Стоян е същият като този на Иван.