Изкуствен интелект / интелигентни системи

гл. асистент д-р Венета Табакова-Комсалова
ФМИ, ПУ „П. Хилендарски“, Пловдив

```
/* програма publication*/
/* Автор - структура author(Собствено име,Фамилия)
Книга - структура book(Заглавие,Автор,Брой страници)
Публикация - предикат publication(Книга,Цена)
Националност - предикат nationality(Автор,Град в който е роден)    */
publication(book('Железниятсветилник',author('Димитър','Талев'),498),10.90).
publication(book('Осъденидуши',author('Димитър','Димов'),456),25.00).
publication(book('Илинден',author('Димитър','Талев'),868),14.00).
publication(book('Ветрена мелница',author('Елин','Пелин'),526),25.00).
publication(book('Песента на колелетата',author('Йордан','Йовков'),560),9.00).
publication(book('Да бъде ден',author('Христо','Смирненски'),450),6.00).
publication(book('Поглед зад очилата',author('Атанас','Далчев'),496),9.00).
publication(book('Хъшове',author('Иван','Вазов'),546),25.00).
publication(book('В полите на Витоша',author('Пейо','Яворов'),362),20.00).
publication(book('Немили-Недраги',author('Иван','Вазов'),366),9.00).
publication(book('През води и гори',author('Емилиян','Станев'),144),4.20).
publication(book('Патиланци',author('Ран','Босилек'),272),12.90).
```

```
publication(book('Под Игото',author('Иван','Вазов'),600),10.00).
publication(book('В лунната стая',author('Валери','Петров'),80),17.00).
publication(book('Аз съм българче',author('Иван','Вазов'),80),12.90).
publication(book('Палечко',author('Валери','Петров'),10),10.00).
publication(book('Мамино детенце',author('Любен','Каравелов'),96),4.00).
publication(book('Зайченцето бяло',author('Леда','Милева'),80),12.90).
publication(book('Тютюн',author('Димитър','Димов'),864),40.00).
publication(book('Крадецът на праскови',author('Емилиян','Станев'),112),13.00).
publication(book('Бай Ганьо',author('Алеко','Константинов'),256),5.00).
publication(book('Ян Бибиян',author('Елин','Пелин'),200),4.00).
publication(book('До Чикаго и назад',author('Алеко','Константинов'),120),4.90).
publication(book('Лавина',author('Блага','Димитрова'),344),12.00).
publication(book('Нощем с белите коне',author('Павел','Вежинов'),464),18.00).
```

```
% Националност - предикат nationality(Автор,Град в който е роден)
nationality(author('Димитър','Талев'),'гр.Прилеп').
nationality(author('Димитър','Димов'),'гр.Ловеч').
nationality(author('Елин','Пелин'),'с.Байлово').
nationality(author('Йордан','Йовков'),'гр.Жеравна').
nationality(author('Христо','Смирненски'),'гр.Кукуш').
nationality(author('Атанас','Далчев'),'гр.Солун').
nationality(author('Иван','Вазов'),'гр.Сопот').
nationality(author('Пейо','Яворов'),'гр.Чирпан').
nationality(author('Емилиян','Станев'),'гр.Велико Търново').
nationality(author('Ран','Босилек'),'гр.Габрово').
nationality(author('Валери','Петров'),'гр.София').
nationality(author('Любен','Каравелов'),'гр.Копривщица').
nationality(author('Леда','Милева'),'гр.София').
nationality(author('Алеко','Константинов'),'гр.Свищов').
nationality(author('Блага','Димитрова'),'гр.Бяла Слатина').
nationality(author('Павел','Вежинов'),'гр.София').
```

1.Да се изведът всички книги с цена над 4.90 ?-publication(X,Y),Y>4.90.

2.Да се изведът всички книги с повече от 200 страници:
а) книгите да се изведът с отделни променливи за име,автор и страници;

?-publication(book(X,Y,Z),_),Z>200.

б)книгите да се изведът като цели структури.

?-publication(X,_),X=book(_,_,Z),Z>200,write(X),nl,fail.

3.Да се намери автор с две различни книги.

two_books(X):-publication(book(Y,X,_),_),publication(book(Z,X,_),_),Z\=Y.
?-two_books(X).

4.Да се намери автор с точно една книга.

one_book(X):-publication(book(Y,X,_),_), \+ (publication(book(Z,X,_),_),Z\=Y).
?-one_book(X).

5.Всички двойки различни автори с едно и също първо име.

same_fname(X,Y):-publication(book(_,author(Z,X),_),_), publication(book(_,author(Z,Y),_),_),X\=Y.
%или
same_fname(X,Y):-publication(book(_,X,_),_),publication(book(_,Y,_),_),
X\=Y,X=author(Z,_),Y=author(Z,_).
?-same_fname(X,Y).

6.Заглавията на книги автори родени в гр.София.

?-publication(book(X,Y,_),_),nationality(Y,'гр.София'),write(X),nl,fail.

7.Да се намери най-евтината книга.

?-publication(X,Y),\+ (publication(_,Z),Z<Y).

За да пишем ефективен код в Пролог е желателно да разберем как работи машината за извод на Пролог. Най-общо Пролог ще търси начин да направи една заявка вярна чрез намиране на подходящи стойности за променливите като се консултира с базата данни.

Унификация

Пролог реализира две стъпки за да определи присвояванията на променливите, когато се опитва да реши една заявка. Първата стъпка се нарича **унификация**.

Унификацията винаги се прилага върху двойка терми. Унификацията проверява дали можем да сравним тези два терма, като намерим конкретни стойности за променливите.

Обръщам внимание, че два терма могат да се унифицират, дори ако имат стойност false. Така напр., жена(стоян) не е верен според примерната база данни, но може да се унифицира с жена(X).

Унификация

Да разгледаме формално процеса на унификация. Нека са дадени два терма T1 и T2. Пролог прави унификация по следния начин:

- Ако T1 и T2 са атоми или числа, тогава те се унифицират когато са идентични;
- Ако T1 е свободна променлива, тогава тези терми се унифицират като T1 се свързва с T2;
- Ако T2 е свободна променлива, тогава се унифицират като T2 се свързва с T1;
- **Ако T1 и T2 са структури, тогава те се унифицират когато:**
  - T1 и T2 притежават еднакви имена;
  - Имат еднакъв брой аргументи;
  - Съответните двойки аргументи унифицират.

За да демонстрираме унификацията и резолюцията в Пролог, ще разгледаме отново примерната база данни на сюжета от „Железният светилник“.

```
родител(стоян,кочо).
родител(стоян,лазар).
родител(стоян,манда).
родител(стоян,нона).
родител(стоян,катерина).
мъж(стоян).
мъж(кочо).
мъж(лазар).
жена(султана).
жена(манда).
жена(нона).
жена(катерина).
родител(султана,кочо).
родител(султана,лазар).
родител(султана,манда).
родител(султана,нона).
родител(султана,катерина).
баща(X) :- родител(X, _), мъж(X).
майка(X) :- родител(X, _), жена(X).
сестра(X,Y):-родител(Z,X), родител(Z,Y),
                жена(X), жена(Y), X \== Y.
```

Както споменахме унификацията е процес на търсене на съвпадение между два предиката и обвързването на техните променливи. Този механизъм е необходим на Пролог да идентифицира коя клауза е обработена и да обвърже със стойност променливите. Съществуват няколко важни момента при осъществяването на унификация:

- Когато Пролог се опита да удовлетвори целта, търсенето на съвпадение започва от началото на програмата.
- Когато се прави ново обръщение за удовлетворяване, търсенето на съвпадение отново започва от началото на програмата.
- Когато при търсене на съвпадение се стигне до успех, започва да се търси решение за следващата подцел.
- Когато една променлива бъде обвързана в клауза единствения начин за нейното освобождаване е чрез механизма на възврат.

Унификация на сложни обекти

Сложните обекти могат да се унифицират както с отделна променлива, така и със сложен обект. Това означава, че е възможно да се използва сложния обект като едно цяло, а също да се използва унификация на отделните елементи. Например: date(14,2,2023). Ако се съпостави с отделна променлива X на този обект, то като цяло променливата X ще се обвърже са date(14,2,2023). От друга страна е възможно date(14,2,2023) да се съпостави на date(DD,MM,YYYY) и по този начин обвързването на променливите да е следното:

DD с 14, MM с 2 и YY с 2023.

Използване на знак за равенство за унификация на сложни обекти

Пролог прилага унификация по два начина. Първият е при **удовлетворяване на цел с глава на клауза**. Вторият начин **преминава през знака за равенство**, който на практика е инфиксен предикат, т.е. предикат, който се намира между аргументите си, а не преди тях. Пролог извършва необходимите съпоставяния на обектите от двете страни на равенството. Това се използва за намиране на стойностите на аргументи в сложен тип обект.

Унифициращ алгоритъм. Пример.

Терми. Съставни терми (структури) – представят съставни обекти (не функция, която връща резултат). Записване: $f(a_1, a_2, a_3, \ldots\ldots, a_n)$; където f е функтор, а термите $a_1, a_2, a_3, \ldots\ldots, a_n$ са аргументи. Терми са също променливите и константите.

Пример 1: owns(tom,horse(blackly)). В този пример структурата е horse(blackly) и представя кон с неговото име. Horse е функтора, а името blackly е аргумента.

Пример 2: owns(tom, car(mercedes, red)). Структурата car(mercedes, red) представя вида и цвета на колата. Функтурът е car, а аргументите са два – mercedes и red.

Процесът на унификация на терми:

- *Константа* (атомен терм) се унифицира с равна на нея константа, свободна променлива или свързана променлива стойност, равна на константата;

- *Променлива* – ако е свободна се унифицира с всеки терм, ако е свързана се унифицира по правилото за константи или съставни терми, в зависимост от стойността ѝ;

- *Съставен терм* се унифицира само с такъв съставен терм, който има същия функтор и същия брой аргументи, и на който аргументите се унифицират със съответните аргументи на дадения терм.

Унификацията служи за:

- Присвояване (свързване на променлива със стойност);
- Предаване на параметри;
- Проверка на равенство;
- Достъп до структури от данни и техните елементи.

Пример: Нека са дадени предикатите:
book(title, pages).
written_by(author, title).
long_novel(title).
И следната програма:

```
written_by(димитър_талев, железният_светилник).
written_by(елин_пелин, гераците).
book(гераците,112 ).
book(железният_светилник, 498).
long_novel(Title):-
    written_by(_,Title),
    book(Title,Length),
    Length > 300.
```

Нека проследим процеса при вземане на решения от Пролог.

Първа цел: written_by(X, Y).

Когато Пролог се опитва да удовлетвори целта written_by(X, Y). е необходимо да се тестват всички клаузи в програмата за съвпадение. При опита за съвпадение на променливите X и Y с аргументите на всяка written_by клауза, Пролог ще започне търсенето от началото на програмата към нейния край.

written_by(X , Y).

written_by(димитър_талев, железният_светилник)

Пролог намира съвпадение и X обвързва с димитър_талев, а Y се обвързва с железният_светилник, и извежда следния резултат:

X=димитър_талев, Y=железният_светилник

Тъй като се проверява за всички възможни решения на целта, Пролог унифицира свободните променливи и с втората клауза на written_by:

written_by(елин_пелин, гераците).

След това извежда и второто решение:

X= елин_пелин, Y=гераците


Program
1 written_by(димитър_талев,железният_светилник).
2 written_by(елин_пелин,гераците).
3 book(гераците,112).
4 book(железният_светилник, 498).
5 long_novel(Title):- written(_,Title),book(Title,Length),Length > 300.
6
written_by(X, Y).
X = димитър_талев,
Y = железният_светилник
X = елин_пелин,
Y = гераците
?- written_by(X, Y).

Втора цел: written_by (X, гераците).

Пролог ще опита да намери съвпадение с първата клауза written_by:

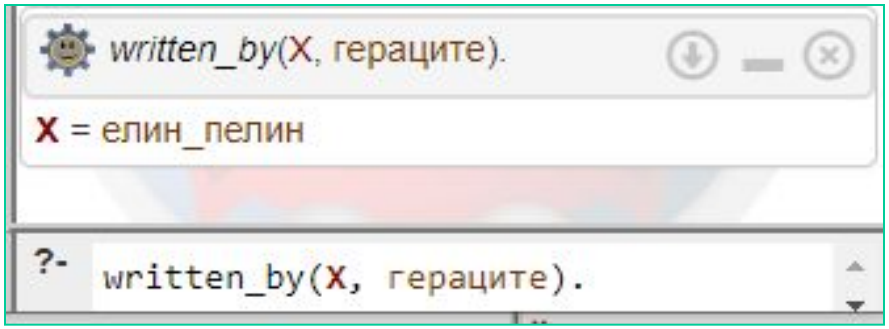
written_by (X , гераците).

written_by(димитър_талев, железният_светилник).

Тъй като „гераците“ и „железният_светилник“ не си съвпадат, опитът за унификация пропада. Пролог ще продължи със следващите факти в програмата:

written_by(елин_пелин, гераците).

Тук ще се намери унификация и X ще се обвърже с „елин_пелин“.



Трета цел: long_novel(X).

Пролог се опитва да удовлетвори целта, като се изследва дали целта не съвпада с някой факт или с глава на правило.

В този случай се намира съвпадение с: long_novel(Title)

Целта съвпада с глава на правило и унификацията е направена. Пролог ще прави опит за удовлетворяване на подцелите на това правило.

long_novel(Title):- written_by(_, Title), book(Title, Length), Length>300.

Удовлетворяването на тялото на правилото ще премине първо през първата подцел written_by(_,Title). Трябва да се обърне внимание, че авторът на книгата не е от значение, тъй като на мястото на author е поставена анонимна променлива.

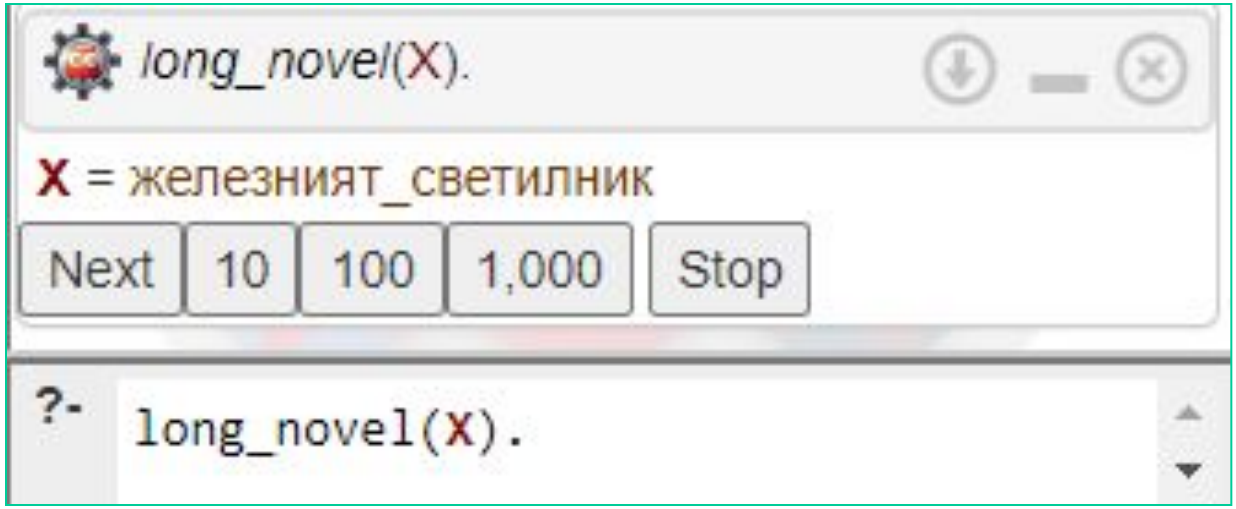
written_by(_ , Title),

(димитър_талев, железният_светилник).

Променливата Title вече се обвързва с „железният_светилник“ и за следващата подцел book(Title,Length) това ще има значение.

Третата клауза в тялото на long_novel сега се превръща в текущата подцел: Length > 300.

Пролог извършва сравнението и успява, тъй като 498 е по-голямо от 300. В този момент всички подцели на правилото са удовлетворени и като цяло целта long_novel(X) успява. Тъй като променливата X от целта се унифицира с променливата Title от правилото, стойността с която правилото успее ще се върне и унифицира с променливата X. Title има стойност „железният светилник“, когато правилото е удовлетворено и поради тази причина изхода на Пролог ще е: X=железният_светилник.



При деклариране на твърденията(предикатите) в програмата името на предиката трябва да започва с малка буква, последвано от буква, цифра или символ за подчертаване с обща дължина от не повече 250 символа.

В името на предиката са забранени използването на интервали, знак минус, звездички, наклонени черти. Стандартния начин за деклариране е следния:
predicateName(argument_type1,..., argument_typeN),

където argument_type1, ..., argument_typeN са или стандартни типове или декларирани.

Възможно е да се запишат няколко действия за един и същ предикат в клауза. Стандартния начин за деклариране е следния:

HEAD :- <Subgoal1>, <Subgoal2>, ..., <SubgoalN>.

|**?** е секцията за определяне на целта. Тук задължително се използва един или няколко от описаните предикати. Възможно е в предикатите да се използват променливи или константи, посочени в необходимия вид.

Вградени предикати:

- **! (отсичане)** – предикатът винаги успява. Използва се за предотвратяване на връщането назад за търсене на други решения или търсенето на решение по друга клауза.
- **not p** – успява, ако p не е истина. Аргументът не може да съдържа свободни променливи.
- **\+ p** – успява, ако p не успее (p не е истина). Аргументът може да съдържа свободни променливи. **Внимание**: Свободните променливи, участващи в аргумент на \+ не могат да получават стойност!
- **Write (<аргумент>)** – отпечатва аргумента без да минава на нов ред.
- **nl –** преминаване на нов ред.
- **fail** – винаги пропада. Използва се когато искаме да предизвикаме връщане назад. Понеже цел, в която участва fail винаги пропада, трябва ние сами да се погрижим да отпечатаме интересуващите ни стойности, получени като решение на подцелите преди **fail.**

ВГРАДЕНИ ПРЕДИКАТИ С ИНФИКСЕН ЗАПИС

Разлика между =, \=, is, =:=, =\=, ==, \==

терм1 = терм2 – (проверка за) унификация на терм1 и терм2;
терм1\=терм2 – проверка за неуфикация (истина е, ако не се унифицират двата терма);
ар.израз 1 =:= ар.израз2 – сравняване на аритметични изрази за равенство, като аритмитичните изрази от двете страни се изчисляват;
ар.израз1 =\= ар.израз2 – сравняване на аритметични изрази за неравенство;
пром. is ар.израз – свързване на свободна променлива със стойността (изчислява се стойността на израза);
променливата може да е и свободна – тогава се сравняват стойностите от двете страни;
терм1 == терм2 – проверява за идентичност (съвпадение);
терм1 \== терм2 – проверява за неидентичност.

В Пролог, както вече говорихме, има константи- числа, които се изписват по обичайния начин. За работа с числа има следните вградени предикати, които имат инфиксен запис:

= - това равенство е малко по особено, то се нарича равенство по унифицируемост, т. е. то може да се прилага не само към числа, но и към променливи и структури. Първо се опитва да унифицира лявата и дясната част и ако успее, отговаря с yes, ako не успее, отговаря с no, като ако при тази унификация някоя променлива се остойности или съвмести, тя си остава такава.

=:= - това е тъй нареченото силно равенство, то успява само в случай, че от ляво и от дясно стоят равни числа, като при това от някоя страна, ако стои остойностена променлива, тя се замества с нейната стойност, ако стои аритметичен израз, той се пресмята. Внимание: ако някъде има свободна променлива, ще пропадне.

\= - това е съответното неравенство на равенството по унифицируемост, то успява, ако = пропадне и обратното. Тук не настъпват никакви остойностявания на променливите.

=\= - това е съответното неравенство на =:=, пресмята изразите от двете страни и ги сравнява.

<,>,=<,>= - имат ясен смисъл, като отново ако от ляво или от дясно стои израз, той първо го пресмята и после ги сравнява. Като при това има следната особеност в SP, ако в някоя от страните стои единствено неустойностена променлива, това дава yes, защото в дефиницията им е използвано =. Ако тази променлива е в израз, вече работи ОК, но това е в случая с =, ако е строго вече дава no при свободна променлива от едната страна, така че трябва да знате да използват тези предикати само със остойностени променливи. Също така трябва да се каже, че стандартът изисква тези предикати да работят само с числа, т. е. в стандарта ако някъде има израз, пропада.

is - Това е Прологовото присвояване, по правило от ляво трябва да стои променлива, а от дясно числов израз, който може да се пресметне, т. е. променливите в него трябва да са свързани;
Пролог пресмята стойността на израза и остойностява променливата от ляво с нея и успява, ако тази променлива е свободна, а ако не е, я сравнява с нейната стойност. В SP това работи и наобратно, т. е. ако от дясно е променливата, а от ляво изразът, успява когато и от двете страни са свободни променливи, като при това ги съвместява,
ако вече в някоя от страните има свободна променлива, която участва в израз, не успява.
Общо взето, принципът явно е: смята и ако не може да сметне нещо, дава no, иначе продължава.

Използване на предиката fail

Пролог използва механизма на възврат, когато търсенето на съвпадение пропадне. В някои ситуации е необходимо да се провокира механизма на възврат, за да се намерят различни решения. В Пролог е предвиден специален предикат **fail,** при който проверката винаги пропада и по този начин се стартира ново търсене на решение. Със следващият пример ще се илюстрира действието на този специален предикат:

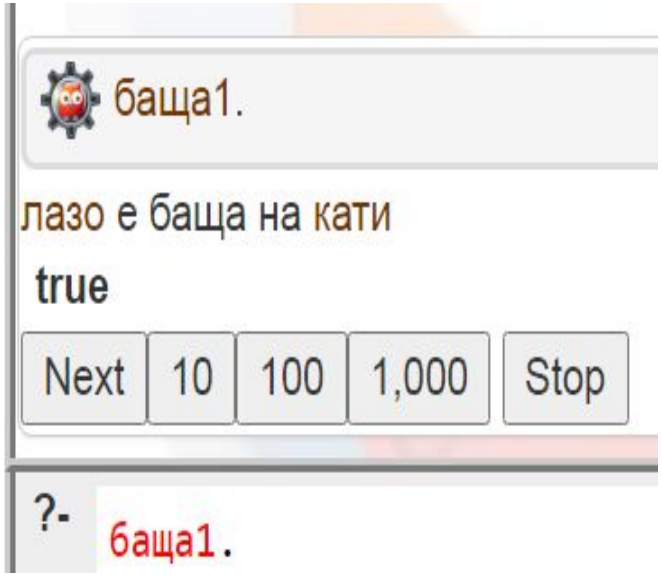
```
баща(лазо,кати).
баща(камен,георги).
баща(камен,мария).
|?- баща(X,Y).
```

След стартирането на програмата, поради заложения механизъм на възврат, ще се получи следното решение:

X=лазо, Y=кати
X=камен, Y=георги
X=камен, Y=мария
3 Solutions

Много по-удачно ще бъде, ако чрез изхода на програмата се разбере реално съществуващата връзка баща, т.е. да се изведе, че „Лазо е баща на Кати”. За целта в програмата ще има необходимост от нов предикат, а именно:

```
баща(лазо,кати).
баща(камен,георги).
баща(камен,мария).
баща1 :- баща(X,Y),write(X),
    write(" е баща на "), write(Y).
баща1.
```



След стартирането на програмата ще извежда всички възможни решения едно по едно. Всяко следващо решение ще го дава след натискане на Next.

За да се изведе само едно решение трябва да стартираме следната програма:

```
баща(лазо,кати).
баща(камен,георги).
баща(камен,мария).
баща1 :-
баща(X,Y),!,write(X),write(" е баща на "),write(Y),nl, fail.
```

```
?- баща1.
```

След стартирането на програмата вместо извеждането на всички възможни решения Пролог извежда само едно.

```
лазо е баща на кати
false
```

Причината за това да не се изведат всички, а само едно решение е, че Пролог е намерил едно решение и нищо не е в състояние да го провокира за търсене на ново. За да може все пак да се изведат всички възможни решения е необходимо да се провокира изкуствено механизма на възврат. Решението е следното:

```
баща(лазо,кати).
баща(камен,георги).
баща(камен,мария).
баща1:- баща(X,Y), write(X),write(" е баща на "), write(Y),nl, fail.
```

|?- **баща1.**

Резултатът вече е :

```
лазо е баща на кати
камен е баща на георги
камен е баща на мария
false
```

```
баща(лазо,кати).
баща(камен,георги).
баща(камен,мария).
баща1 :- баща(X,Y),write(X),
    write(" е баща на "), write(Y),nl, fail.
```

баща1.

лазо е баща на кати
камен е баща на георги
камен е баща на мария
false

?- баща1.

```
баща(лазо,кати).
баща(камен,георги).
баща(камен,мария).
баща1 :- баща(X,Y),write(X),
    write(" е баща на "), write(Y),nl, fail.
баща1.
```

баща1.

лазо е баща на кати
камен е баща на георги
камен е баща на мария
true

?- баща1.

Талисманите на месеците

По древно поверие всеки месец има свой камък-талисман. Например, месеците юни, юли и септември съответстват на камъните рубин, сапфир и перла. Тези камъни означават мъдрост, здраве и благополучие. На кой месец какъв камък-талисман съответства и какво означава, ако се знае, че:

- перла и рубин не съответстват на септември;
- през юни и юли мъдрост не се наблюдава;
- здраве не съответства на рубин;
- благоденствие не се отнася за месец юни.

Лекари:

Дадена е база данни на Пролог от следния вид:

```
% работа(лекар, лечебно_заведение, трудов_стаж)
работа('Иван Петров', 'Трета поликлиника', 23).
работа('Стоян Драганов', 'Пълмед', 8).
работа('Михаела Тодорова', 'Първа поликлиника', 14).
%.... Допишете още факти по посочения начин

% лекува(лекар, пациент) – описващ отношението лекар-пациент и пациент(име, възраст)
лекува('Михаела Тодорова', пациент('Иван Иванов', 34)).
лекува('Стоян Драганов', пациент('Нели Томова', 53)).
лекува('Иван Петров',пациент('Христо Ангелов', 76)).
лекува('Иван Петров',пациент('Камен Димов', 23)).
%.... Допишете още факти по посочения начин
```

А) Дефинирайте предикат, който определя лекарите с трудов стаж 10 години.
Б) Напишете цел, която извежда като резултати пациенти на възраст 34 г., лекувани от д-р Михаела Тодорова.
В) Дефинирайте предикат, определящ имената на пациентите, които се лекуват при лекар работещ в Трета поликлиника и има стаж под 15 години.
Г) Напишете цел, която извежда двама различни пациенти, които се лекуват при един и същи лекар.
Д) Дефинирайте предикат, който определя пациентите, които не се лекуват в Първа поликлиника.
Е) Напишете цел, която извежда пациентите, които се лекуват само при един лекар.

А) Дефинирайте предикат, който определя лекарите с трудов стаж 10 години.
Б) Напишете цел, която извежда като резултати пациенти на възраст 34 г., лекувани от д-р Михаела Тодорова.
В) Дефинирайте предикат, определящ имената на пациентите, които се лекуват при лекар работещ в Трета поликлиника и има стаж под 15 години.
Г) Напишете цел, която извежда двама различни пациенти, които се лекуват при един и същи лекар.
Д) Дефинирайте предикат, който определя пациентите, които не се лекуват в Първа поликлиника.
Е) Напишете цел, която извежда пациентите, които се лекуват само при един лекар.

```
стаж_10(A):-работа(A,_,C),C<10, write(A), nl, fail.
пациент_3(A):-лекува(C,A),работа(C,'Трета поликлиника',B), B > 15.
пациент_не1(A):-лекува(B,пациент(A,_)), \+ работа(B,'Първа поликлиника',_),     write(A), nl, fail.
%.... Допишете още факти по посочения начин
/* А) Дефинирайте предикат,  който определя лекарите с трудов стаж под 10 години.
 стаж10(A):- работа(A,_,B),B<10.
 или
 стаж_10(A):-работа(A,C, B),B<10, write(A), nl, fail.
Б) Напишете цел, която извежда като резултати пациенти на възраст 34 г., лекувани от д-р Михаела
Тодорова.
?- лекува(A, пациент(B,C)), A='Михаела Тодорова',
C=34, write(B), nl, fail.
или
лекува('Михаела Тодорова', пациент(B,34)),
write(B), nl, fail.
В) Дефинирайте предикат, определящ имената на пациентите, които се лекуват при лекар работещ в Трета
поликлиника и има стаж под 15 години.
пациент_3(A):-лекува(C,A),работа(C,'Трета поликлиника',B), B<15.

Г) Напишете цел, която извежда двама различни пациенти,
които се лекуват при един и същи лекар.
?- лекува(A,B), лекува(A,C), B\=C.
Д) Дефинирайте предикат, който определя пациентите, които не се лекуват в Първа поликлиника.
пациент_не1(A):-лекува(B,A), \+работа(B,'Първа поликлиника',_).
или
пациент_не1(A):-лекува(B,A),работа(B,C,_), C\='Първа поликлиника'.

Е) Напишете цел, която извежда пациентите, които се лекуват само при един лекар.
?- лекува(A,B), лекува(C,B), A\=C, write(B), nl, fail.
*/
```