МУЛЬТИПАРАДИГМЕННЕ ПРОГРАМУВАННЯ

Лекція 4

ПРОГРАМУВАННЯ НА МОВІ ПРОЛОГ

БАЗИ ЗНАНЬ

Як ми вже відзначали, програма мовою Пролог, що містить факти й правила, становить базу знань. При розробці програмна Пролозі часто використовують вбудовані предикати, тобто предикати, обумовлені автоматично при ініціалізації інтерпретатора Прологу.

Вбудовані предикати використовуються так само, як й обумовлені користувачем предикати. Єдине обмеження - вбудований предикат не може бути головою правила або з'являтися у факті.

Одним з найчастіше використовуваних вбудованих предикатів є предикат **not** (заперечення). Цей предикат істинний, якщо його аргумент помилковий,навпаки. Можна використати й іншу форму запису даного предиката **\+**.

Приклад

Якщо ми визначимо правило

```
неправда(X) :- not(X).
неправда1(X) :- \+(X).
```

то наступні запити будуть еквівалентні:

```
?- not(більше(собака, кінь)).
Yes;
?- неправда(більше(собака, кінь)).
Yes
```

Іншим часто використовуваним вбудованим предикатом є **=** (уніфікація):
=(X, Y).
Цей предикат допускає більше зручну форму запису **X = Y**. Значення цього предиката істинно, якщо терми **X** й **Y** вдається уніфікувати.

На предикат **not** схожий вбудований предикат **\=**, що залежить від двох аргументів.
Твердження **X \= Y** еквівалентно твердженню **not(X = Y)**.

Іноді буває корисно використати предикати, про які заздалегідь відомо, істинні вони або помилкові.
Для цих цілей використовують предикати **true** й **fail**. Предикат true завжди істинний, у той час як fail завжди помилковий.

Вбудований предикат **read** дозволяє зчитувати терми із клавіатури. При цьому запрошення Прологу **?-** міняється на **|:**. Терм, що вводиться, повинен обов'язково закінчуватися крапкою.

Приклад

```
?- read(Name), read(Age).
|: микола. 15.
Name = микола
Age = 15
Yes
```

```
?- read(X), більше_2(X,Y).
|: віслюк.
X = віслюк
Y = собака ;
X = віслюк
Y = мавпа ;
No
```

Якщо при обробці запитів Прологу ви побажаєте одержати більш докладний висновок, то для цих цілей можна використати предикат **write**.
Аргументом цього предиката може бути будь-який припустимий терм Прологу.
У випадку, коли аргументом є змінна, буде надруковане її значення.

У випадку, коли аргументом є змінна, буде надруковане її значення. Виконання предиката **nl** здійснює перенiс рядка: наступне виведення почнеться з нового рядка.
Предикат **tab** виводить кількість пробілів, визначену його аргументом.

Приклад

```
?- write('Hello World!').
Hello World!
Yes
?- write('Hello'), nl, tab(5), write('World!').
Hello
     World!
Yes
```

```
?- X = слон, write(X), nl.
Слон
X = слон
Yes
```

В останньому прикладі спочатку змінна **X** уніфікується з атомом слон, а потім значення змінної **X**, тобто **слон**, виводиться на екран за допомогою предиката **write**. Після переходу на новий рядок Пролог видає звіт про уніфіковану змінну, тобто друкує **X = слон**.

Приклад

```
?- help(write).
write(+Term)
Write Term to the current output, using brackets and
operators
where appropriate. See feature/2 for contrillong floating
point
output format.
write(+Stream, +Term)
Write Term to Stream.
Yes
```

І, наостанок, поговоримо про коментарі. Коментарі ніяк не впливають на виконання програми, але при їх правильному використанні вони являються досить істотною частиною вихідного тексту. Трохи вдало розташованих рядків коментарями дуже допоможуть людині, яка читає програму.

Пролог ігнорує довільне число рядків, укладене між символами **/*** й ***/**.
Усе, що перебуває між **%** і кінцем рядка, теж розглядається як коментар:

Приклад

```
/* Це
коментар */
% Це теж коментар+
```

Рішення логічних задач

Інтерпретатор Прологу можна змусити вирішувати логічні задачі, що недоступно більшості процедурних мов.

Багато логічних задач пов'язані з розглядом декількох кінцевих множин однаковою кількістю елементів, між якими встановлюється взаємооднозначна відповідність.

Мовою Пролог ці множини можна описувати як бази даних,залежності між об'єктами встановлювати за допомогою правил.

Приклад

В автомобільних перегонах три перших місця зайняли Олексій, Петро й Микола. Яке місце зайняв кожний з них, якщо Петро зайняв не друге й не третє місце, а Микола - не третє?

І'мя	***І місце***	***ІІ місце***	***ІІІ місце***
Олексій			
Петро	-	-	
Микола		-	

Традиційним способом задача вирішується заповненням таблиці.умовою задачі Петро зайняв не друге й не третє місце, а Микола - не третє. Це дозволяє поставити символ '-' у відповідних клітках. Між множиною імен учасників перегонів й множиною місць повинне бути встановлене взаємо-однозначна відповідність.

Тому визначаємо зайняте місце спочатку в Петра, потім у Миколи й, нарешті, в Олексія. У відповідних клітках проставляємо знак '+'. У кожному рядку й кожному стовпці повинен бути тільки один такий знак.

І'мя	*І місце*	*ІІ місце*	*ІІІ місце*
Олексій	-	-	+
Петро	+	-	-
Микола	-	+	-

З останньої таблиці випливає, що Олексій посів третє місце, Петро перше, Микола - друге.

Мовою Пролог структура програми буде наступною: спочатку перераховуються дані - імена й номер зайнятого місця, а потім записуються правила, що зв'язують ці дві множини.

```
/* База даних імен */
ім'я(олексій).
ім'я(петро).
ім'я(микола).
/* База даних призових місць */
місце(перше).
місце(друге).
місце(третє).
```

/* Встановлюємо взаємо-однозначну відповідність між базами даних, X - елемент із бази даних імен, Y - елемент із бази даних зайнятих місць */
/* Петро зайняв не друге й не третє місце */
відповідність(X, Y) :- ім'я(X), X=петро, місце(Y), not(Y=друге), not(Y=третє).

/* Микола посів не третє місце */

відповідність(X, Y) :- ім'я(X), X=микола, місце(Y), not(Y=третє).

відповідність(X, Y) :- ім'я(X), X=олексій, місце(Y).

```
/* У всіх хлопців різні місця */
рішення(X1,Y1,X2,Y2,X3,Y3) :-
X1=петро, відповідність(X1,Y1),
X2=микола, відповідність(X2,Y2),
X3=олексій, відповідність(X3,Y3),
Y1\=Y2, Y2\=Y3, Y1\=Y3.
```

Для одержання відповіді варто виконати запит

?- рішення(X1,Y1,X2,Y2,X3,Y3).

У відповідь Пролог видасть імена хлопців і зайняті ними місця. Перевірте, чи є інші варіанти відповідей.

Приклад
Вітя, Юра та Михайло сиділи на лавці. У якому порядку вони сиділи, якщо відомо, що Михайло сидів ліворуч від Юри, а Вітя ліворуч від Михайла. В умові задачі перераховуються об'єкти одного типу, зв'язані між собою.

Заповнимо базу даних:
/* Михайло сидів ліворуч від Юри */
ліворуч(юра, михайло).
/* Вітя сидів ліворуч від Михайла */
ліворуч(михайло, вітя).

Правило для встановлення проходження об'єктів один за одним буде таким:

/* Об'єкти X, Y й Z утворять ряд,
якщо X ліворуч від Y й Y ліворуч від Z */

ряд(X, Y, Z) :- ліворуч(Y, X), ліворуч(Z, Y).

Кількість аргументів у голові даного правила дорівнює кількості об'єктів у задачі. Запит

?- ряд(X, Y, Z).

дасть нам рішення задачі.

Арифметичні вирази

У мові Пролог є ряд вбудованих функцій для обчислення арифметичних виразів, деякі з яких приведені в таблиці нижче.

X + Y	Сума X та Y
X - Y	Різниця X та Y
X * Y	Добуток X та Y
X / Y	Ділення X на Y
X mod Y	Остача від ділення X на Y
X // Y	Ділення націло X на Y
X ** Y	Піднесення X у ступінь Y
- X	Зміна знака X
abs(X)	Абсолютна величина числа X
max(X,Y)	Більше з чисел X та Y
min(X,Y)	Менше з чисел X та Y
sqrt(X)	Квадратний корінь з X

random(Int)	Випадкове ціле число в діапазоні від 0 до Int
sin(X)	Синус X
cos(X)	Косинус X
tan(X)	Тангенс X
log(X)	Натуральний логарифм (ln) числа X
log10(X)	Десятковий логарифм (lg) числа X
float(X)	Дробове число, що відповідає цілому числу X
pi	3.14159 (наближене значення числа π)
e	2 .71828 (наближене значення числа e)

Ще раз відзначимо, що Пролог намагається сховати різницю між арифметикою цілих і дробових чисел скрізь, де це можна. Для обчислення арифметичних виразів у Пролозі використовується вбудований бінарний оператор **is**, що інтерпретує правий терм як арифметичний вираз, після чого уніфікує (якщо можливо) результат обчислення з лівим термом (звичайно зі змінною).

Пріоритет виконання арифметичних операцій є традиційним. Круглі дужки використаються для зміни порядку обчислень.
У наступних прикладах змінна X уніфікується зі значеннями арифметичних виразів:

?- X is 2.5 + 2.5.
X = 5
Yes

```
?- X is 4/(2+1).
X = 1.33333
Yes
?- X is cos(3*pi).
X = -1
Yes
?- 1 is sin(pi/2).
Yes
?- 1.0 is sin(pi/2).
No
```

Пояснимо трохи несподівану відповідь Прологу в останньому запиті. Значення **sin(pi/2)** автоматично округляється предикатом is до цілого значення **1**, що не вдається уніфікувати з дробовими числом **1.0**. Предикат **float** змусить вважати значення **sin(pi/2)** дробовими числом:

```
?- 1.0 is float( sin(pi/2)).
Yes
```

Для порівняння арифметичних виразів використається ряд операторів.
Ціль **X > Y** (більше) буде успішна, якщо вираз **X** буде відповідати більшому числу, ніж вираз **Y**.
Аналогічно використаються оператори **<** (менше), **=<** (менше або дорівнює), **>=** (більше або дорівнює), **=\=** (не дорівнює) і **=:=** (арифметично рівний).

Розходження між операторами **=:=** й **=** дуже істотні. Перший оператор порівнює значення арифметичних виражень, тоді як останній намагається уніфікувати їх.

Приклад

```
?- 2 ** 3 =:= 3 + 5.
Yes
?- 2 ** 3 = 3 + 5.
No
?- 1.0 = float(sin(pi/2)).
No
?- 1.0 =:= sin(pi/2).
Yes
```

Помітьте, що ціль **X =:= Y** буде істинна, навіть якщо один з термів є ціле число, а інший - рівне йому дробове.

Приклад

Порядок підцілей у запиті впливає на його результат:

?- X is 4+Y, Y=3.

ERROR: Arguments are not sufficiently instantiated

?- Y=3, X is 4+Y.

Y = 3

X = 7

Yes

У першому запиті повідомлення про помилку з'явилося тому, що перша підціль запиту **(X is 4+Y)** зазнала невдачі, тому що в момент її обробки неможливо обчислити вираз **4+Y**.

Приклади програм

Розглянемо деякі програми, що демонструють обробку числових даних. Відзначимо важливу особливість процедур, створюваних мовою Пролог: вони, у відмінності від вбудованих функцій, не можуть з'являтися в арифметичних виразах.

Якщо потрібно, наприклад, змінної **R** привласнити значення, рівне помноженому на три більшому із двох виразів **X** й **Y**, то, використовуючи введену нижче процедуру **максимум**, це можна записати так:

максимум(X,Y,Z), R is 3*Z.

```
максимум(X,X,X).
максимум(X,Y,X):-X>Y.
максимум(X,Y,Y):-X<Y.
```

```
гіпотенуза(X,Y,Z):- number(X), number(Y), Z
is sqrt(X**2 + Y**2).
мін_гіп(A1,B1,A2,B2,Min):-
гіпотенуза(A1,B1,C1),
гіпотенуза(A2,B2,C2),
Min is min(C1,C2).
```

```
сума(X,Y):- integer(X), X<10, Y is X.
сума(X,Y):- integer(X), X1 is X//10, сума(X1,Y1),
Z is X mod 10, Y is Y1+Z.

друк_суми:- write('Введіть число (не
забудьте крапку наприкінці): '),
read(X), nl,
write('Сума цифр введеного числа дорівнює
'),
сума(X,Y), write(Y), nl.
```

```
факт(1,1).
факт(N,R):- integer(N), N>1, N1 is N-1,
факт(N1,R1), R is N*R1.

сума_списку([ ],0).
сума_списку([H|T],S):- сума_списку(T,S1),
number(H), S is
S1+H.
```

Дамо пояснення до наведених програм.

Приклад

Написати процедуру, що обчислює максимум із двох чисел.

```
максимум(X,X,X).
максимум(X,Y,X):- X>Y.
максимум(X,Y,Y):- X<Y.
```

У предикаті **максимум** третій аргумент є максимумом з двох чисел – першого та другого його аргументів. Зміст кожного із правил даної процедури цілком очевидний. Подивимося на реакцію інтерпретатора Прологу на запити, що містять даний предикат.

```
?- максимум(20,50,X).
X = 50
Yes
```

```
?- максимум(100,50,X).
X = 100
Yes
?- максимум(X,50,100).
X = 100
Yes
```

Остання відповідь показує, що наш предикат дозволяє знаходити відповідь на питання типу: "Яке повинне бути число, щоб максимум із шуканого числа й числа 50 дорівнювався 100?".

Як ви думаєте, чому була отримана відповідь "No" на наступний запит?

?- максимум(X,50,40).
No

Приклад

Складіть процедуру **гіпотенуза**, що по двох катетах прямокутного трикутника обчислює його гіпотенузу.

Скористаємося теоремою Піфагора та вбудованою функцією **sqrt** для обчислення квадратного кореня:

```
гіпотенуза(X,Y,Z):- Z is sqrt(X**2 + Y**2).
```

Програма коректно обчислює гіпотенузу, але якщо ми спробуємо за її допомогою знайти один з катетів, то переконаємося, що процедура працює не цілком вірно. Щоб уникнути цього додамо перевіркутого, що перші два аргументи предиката - позитивні числа, для чого використаємо вбудований предикат **number** і порівняння з нулем:

```
гіпотенуза(X,Y,Z):- number(X), X>0, number(Y), Y>0,
                    Z is sqrt(X**2 + Y**2).
```

?- гіпотенуза(3,4,X).
X = 5
Yes

?- гіпотенуза(3,'a',X).
No

?- гіпотенуза(3,X,5).
No

Приклад

Напишіть предикат, що по двох парах чисел - довжинам катетів прямокутних трикутників - визначає величину меншої з гіпотенуз. Скористаймося процедурою **гіпотенуза**, яка розібрана вище, та вбудованою функцією **min**:

```
мін_гіп(A1,B1,A2,B2,Min):-
гіпотенуза(A1,B1,C1),
гіпотенуза(A2,B2,C2),
Min is min(C1,C2).
```

Запити до інтерпретатора Прологу можуть виглядати так:

```
?- мін_гіп(3,4,8,6,X).
X = 5
Yes
?- мін_гіп(3,4,Y,6,X).
No
```

Приклад

Факторіалом натурального числа n є добуток всіх цілих чисел від 1 до n включно. Для запису факторіала числа n використають позначення n!.

n!=n*(n-1)*(n-2)*...*2*1=n*(n-1)!

Наступна процедура обчислює факторіал числа. Зверніть увагу на використання **рекурсії** в даній процедурі:

```
факторіал(1,1).
факторіал(N,R):- integer(N), N>1, N1 is N-1,
факторіал(N1,R1), R is N*R1.
```

Перше правило (так званий термінальний випадок, тобто той момент виконання процедури, коли вона перестає викликати сама себе) говорить, що факторіал одиниці дорівнює одиниці. Друге правило є просто запис визначення факторіала: результат R виходить множенням числа N на факторіал числа, яке на одиницю менше. Воно буде спрацьовувати при всіх n>1 тому, що інтерпретатор Прологу переглядає базу даних зверху вниз і переходить до наступного правила або факту тільки в тому випадку, коли він не може виконати поточне правило.

Приклад

Напишіть програму мовою Пролог, що друкує суму всіх цифр введеного з клавіатури числа.

Для рішення даної задачі скористаємося двома предикатами. Предикат **сума** має своїм першим аргументом число, сума цифр якого є його другим аргументом. Другий предикат - **друк_суми** - запитує число, викликає предикат **сума** та друкує отриманий результат.

```
сума(X,Y):- integer(X), X<10, Y is X.
сума(X,Y):- integer(X), X1 is X//10, сума(X1,Y1),
Z is X mod 10, Y is Y1+Z.

друк_суми:- write('Введіть число (наприкінці крапка): '),
read(X), nl, сума(X,Y),
write('Сума цифр числа '), write(X),
write(' дорівнює '), write(Y), nl.
```

Правило **друк_суми** не має аргументів, дані вводяться з клавіатури й потім, за допомогою механізму уніфікації, передаються іншим підцілям даного правила.

Приклад

Напишіть програму мовою Пролог, що вводить з клавіатури два числа - координати точки на площині та визначає, чи попадає дана точка в коло одиничного радіуса із центром на початку координат.

```
inside(X,Y,попадає):- number(X), number(Y),
                      X**2+Y**2=<1.
inside(X,Y,не_попадає):-number(X), number(Y),
                        X**2+Y**2>1.

 /* Ввести два числа та викликати предикат inside */

 input:-write('Введіть x-координату: '),
        read(X), nl,
        write('Введіть y-координату: '),
        read(Y), nl,
        inside(X,Y,R), write(R).
```