Лабораторна работа №5

Тема роботи : Ознайомлення з принципами організації і експлуатації базової Prolog-системи.

Мета роботи : ознайомлення з основними можливостями системи SWI-Prolog, правилами експлуатації інтерпретатора і заповнення інформаційної бази, складання елементарних і складених питань із стандартними системними предикатами і с складанням простих правил для їх накопичення в Prolog-системі.

1. Ознайомлення з середовищем розробки

a. Установка середовища розробки

Середовище розробки SWI-Prolog можна безкоштовно завантажити з офіційного сайту http://www.swi-prolog.org/. Середовище розробки складається з виконавчого модуля (SWI-Prolog - з ним можна працювати окремо через командний рядок) і графічного інтерфейсу розробника (SWI-Prolog Editor), який є надбудовою над виконавчим модулем.

Завантажити останню версію середовища програмування **SWI-Prolog - Editor** для MS Windows можна за адресою http://lakk.bildung.hessen.de/netzwerk/faecher/informatik/swiprolog/indexe.html. Інтерфейс середовища програмування підтримує декілька мов, включаючи російський. Установка програми виконується за стандартною процедурою.

Також для коректної роботи Help необхідно викачати, розархівувати і помістити усі файли документації в теку "%Program files%\Manual\" (теку Manual необхідно створити).

Після установки і запуску SWI-Prolog - Editor необхідно вибрати пункт меню «Вікно / Конфігурація», перейти на вкладку «Налаштування» і в полі «Codepage» (англ. кодова сторінка) вказати «1251».

За умовчанням в програмі виставлена англійська мова, але його можна змінити на російський, за допомогою Window ->Configuration, де вибрати вкладку Option і в полі Language File вказати шлях до файлу russian.ini (зазвичай поставляється разом з Prolog Editor і знаходиться в тій же теці, що і english.ini).

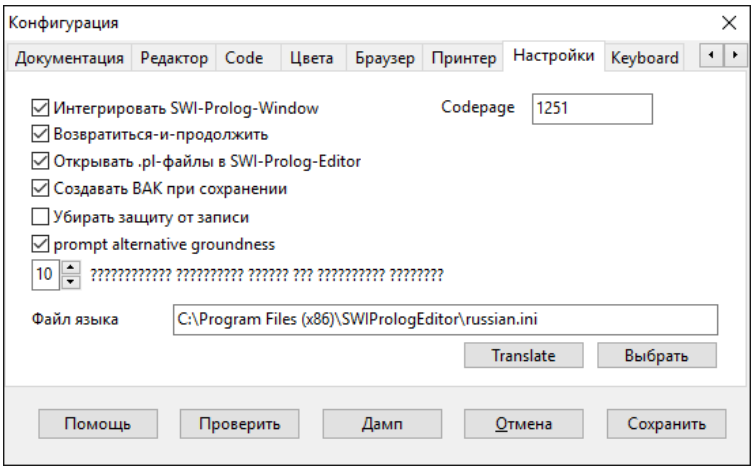
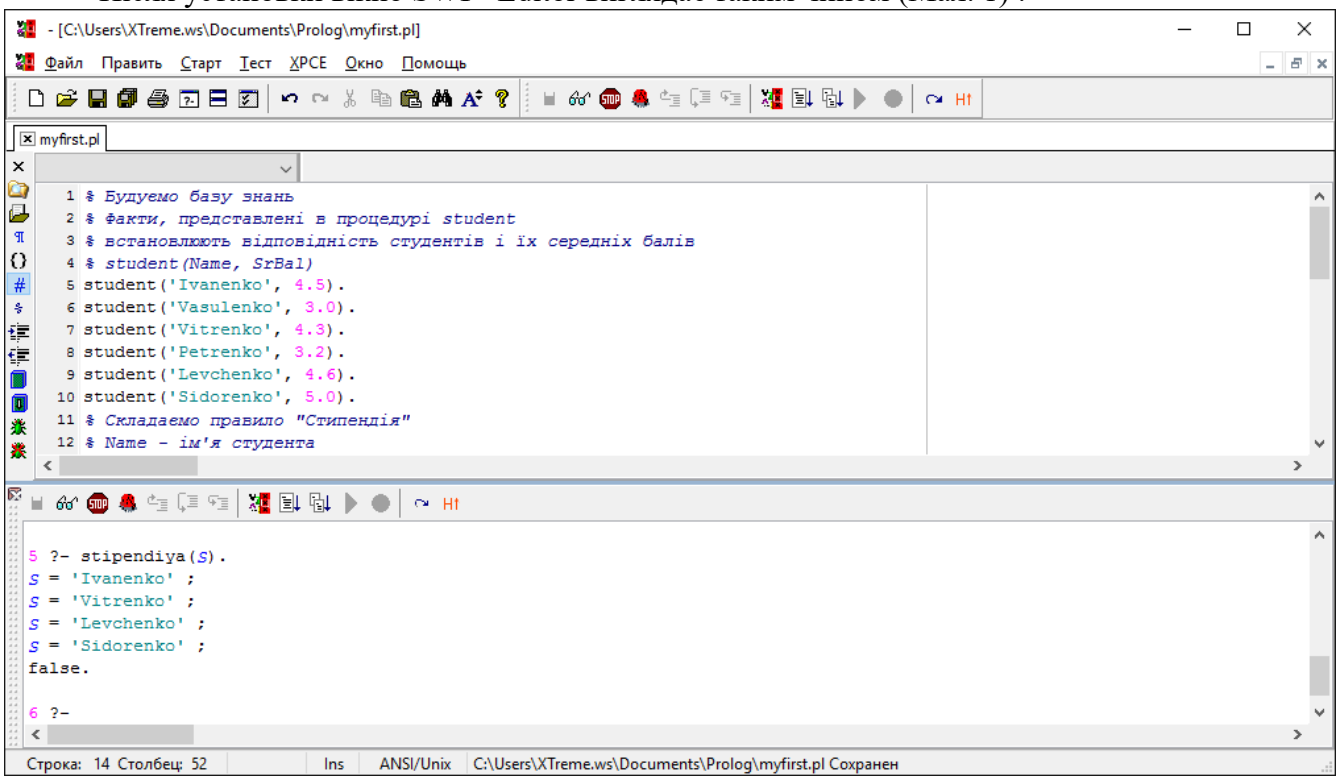


Рис.1. Налаштування кодової сторінки

Налаштування кодової сторінки потрібне для правильного зіставлення рядкових констант, набраних кириличним алфавітом, між текстом програми в середовищі SWI-Prolog - Editor і мовою SWI-Prolog.

b. Основні можливості середовища розробки

Після установки вікно SWI - Editor виглядає таким чином (Мал. 1) :



Мал. 1 Вікно SWI- ditor

Далі пояснення по інтерфейсу редактора будуть викладені з назвами пунктів російською мовою. Майже усі пункти контекстного меню за умовчанням для зручності винесені на верхню панель інструментів.

Для створення нової програми необхідно вибрати пункт Файл->Новий. Після написання програми у вікні редактора її можна виконати командою Старт->Запустити, або натиснути F9. УВАГА : редактор не підтримує кирилицю в шляху до файлу, тому для успішного запуску програм необхідно щоб в шляху до файлу програми не було символів кирилиці. Після будь-якої модифікації програму вимагається наново завантажити в пам'ять. У разі відсутності помилок в програмі, в панелі запитів буде видано повідомлення «true».

У редакторі є можливість відлагодження програм. Для цього необхідно розставити точки зупинки в програмі. Точки зупинки можна поставити клацнувши двічі на цифрі з номером рядка, а також натисненням на кнопку «жучка» на лівій панелі інструментів (не плутати із закладками, закладки - прямокутні, точки зупинки - у вигляді жучків. Закладки ставляться просто для відмітки рядка також подвійним клацанням, але зліва від цифри з номером рядка). Режим відладки включається і відключається пунктом Тест->Відладка (On/Off). Також можливе включення і відключення покрокового виконання програми пунктом Тест->Трасування (On/Off). У режимі покрокового виконання для переміщення між командами використовуються кнопки «Наступний» (наступний функтор), «Крок» (виконання одного кроку програми) і «Крок вгору» (виконання до виходу на рівень вгору по стеку виклику).

У командному рядку відображаються усі дії SWI-Editor, які стосуються движка SWI-Prolog. Насправді SWI-Editor просто дає команди на виконання через командний рядок, тобто є надбудовою над консольною реалізацією середовища розробки SWI Prolog.

2. Теоретичні відомості

a. Загальна інформація

Мова Пролог, найвідоміша з представників сімейства мов логічного програмування, бере початок з робіт Алана Колмерауэра (A. Colmerauer) по обробці природної мови і незалежних робіт Роберта Ковалького (R. Kowalski) по додатках логіки до програмування. Девіду

Уоррену (D. Warren) і його колегам з Едінбурзького університету вдалося здійснити досить ефективну реалізацію Прологу. Ім'я Уоррена увійшло до історії логічного програмування. У його честь названа базова техніка реалізації Прологу, що дістала назву *абстрактної машини Уоррена. Програма на мові Пролог є набором фактів і (можливо) правил. Якщо програма містить тільки факти, то її називають база даних. Якщо вона містить ще і правила, то часто використовують термін база знань*.

SWI-Prolog поширюється під ліцензією GPL, що забезпечує можливість його використання без порушень чиїх-небудь комерційних інтересів. Ця версія мови Пролог доступна як користувачам ОС Linux, так і користувачам Windows.

b. Атоми, числа, змінні і складені терми

Програма на мові Пролог зазвичай описує деяку сутність. Об'єкти (елементи) описуваного світу представляються за допомогою термів. **Терм інтуїтивно означає** *об'єкт. Існує 4 види термів : атоми, числа, змінні і складені терми. Атоми і числа іноді групують разом і називають простими термами*.

Атом - це окремий об'єкт, що вважається елементарним. У Пролозі атом представляється послідовністю букв нижнього і верхнього регістра, цифр і символу підкреслення '_', що розпочинається з рядкової букви. Крім того, будь-який набір допустимих символів, поміщений в апострофи, також є атомом. Нарешті, комбінації спеціальних символів + - * = < > : & також є атомами (слід зазначити, що набір цих символів може відрізнятися в різних версіях Прологу). Приклад

Представлені далі послідовності є коректними атомами:

b, abcXYZ, x_123, efg_hij, Коля, слюсар

'Це також атом Прологу'

+, ::, <---->, ***

Числа в Пролозі бувають цілими (Integer) і речовими (Float). Синтаксис цілих чисел простий, як це видно з наступних прикладів: 1, 1313, 0, - 97. Не усі цілі числа можуть бути представлені в машині, їх діапазон обмежений інтервалом між деякими мінімальним і максимальним значеннями, визначеними конкретною реалізацією Прологу. SWI-Prolog допускає використання цілих чисел в діапазоні від -2147483648 (-2^{31}) до 2147483647 (2^{31}-1).Синтаксис дійсних чисел також залежить від конкретної реалізації. Ми дотримуватимемося простих правил, зрозумілих з наступних прикладів : 3.14, -0.0035, 100.2. При звичайному програмуванні на Пролозі дійсні числа використовуються рідко. Причина цього криється в тому, що Пролог - мова, призначена в першу чергу для обробки символьної, а не числової інформації. При символьній обробці часто використовуються цілі числа, потреба ж в дійсних числах невелика. Скрізь, де можна, Пролог намагається привести число до цілого виду.

Змінними в Пролозі є рядки символів, цифр і символу підкреслення, що розпочинаються із заголовної букви або символу підкреслення :

X, _4711, X_1_2, Результат, _x23, Объект2, _

Останній приклад (єдиний символ підкреслення) є особливим випадком - *анонімною змінною* (змінній без імені). Анонімна змінна застосовується, коли її значення не використовується в програмі. Можливе неодноразове вживання безіменної змінної в одному вираженні застосовується для того, щоб підкреслити наявність змінних за відсутності їх специфічної значущості.

Складені терми (*функції*) складаються з імені функції (нечислового атома) і списку аргументів (термів Прологу, тобто атомів, чисел, змінних або інших складених термів), поміщених в круглі дужки і розділених комами. Групи складених термів використовують для складання фраз Прологу. Не можна поміщати символ пропуску між функтором (ім'ям функції) і відкриваючою круглою дужкою. У інших позиціях, проте, пропуски можуть бути корисні для створення більше читаних програм. Нижче приведені два складені терми:

разом(клієнт(X, 23,_), 71)

'Що сталося?'(нічого)

При завданні імен термів прийнятніше використати мнемонічні (такі що "говорять") імена, оскільки терм а(ж), наприклад, набагато менш інформативний, чим терм автор(жюль_верн).

Ще однією важливою структурою даних в Пролозі є список. Ми познайомимося з ним пізніше. Зараз відмітимо тільки один з видів списків - список символів. Такі списки можуть бути представлені у вигляді **рядків**, наприклад, перший аргумент складеного терма вік("Борис", 10) - рядок. При записі рядки беруть в лапки.

c. Запити, факти і правила

Запит - це послідовність предикатів, розділених комами і завершується крапкою. На природній мові кома відповідає сполучнику "і", а на мові математичної логіки означає кон'юнкцію. За допомогою запитів можна "запитувати" базу даних про те, які твердження є істинними. Предикат запиту називається метою. Прості питання, що не містять ніяких змінних, називають так-ні-питаннями. Вони допускають лише дві можливі відповіді: "Yes" означає наявність відповідного факту у базі даних (перший запит прикладу, приведеного нижче), "No" - його відсутність (другий запит). У разі відповіді "Yes" говорять, що запит завершився успіхом, мета досягнута.

Приклад

```
?- більше(слон, кінь), більше(кінь, осел).
Yes

?- більше(слон, собака).
No
```

Використання *змінних* в запитах дозволяє ставити складніші питання. Припустимо, наприклад, що ми хочемо визначити, які тварини більше осла? У наступному запиті змінна X означає шукану відповідь:

```
?- більше(X, осел).
X = кінь
Yes
```

При обробці запиту змінна X набула значення "кінь". Переглядаючи базу даних, інтерпретатор виявив факт, що стверджує, що кінь більше осла, і запит був успішно виконаний.

Запити зі змінними можуть мати більше за одне рішення. Першим завжди виводиться те з рішень, яке знаходиться ближче до початку бази даних. Якщо нам досить тільки однієї відповіді, то можна натиснути Enter і закінчити пошук. У разі, якщо ми захочемо отримати чергову відповідь, треба натиснути клавішу ; (крапка з комою) і Пролог почне пошук інших варіантів відповіді на запит. Повідомлення "No" говорить про відсутність чергового рішення.

Факт - це твердження про те, що дотримується деяке конкретне відношення. Він є безумовно вірним. У розмовній мові під фактом розуміється щось на зразок "Сьогодні сонячно" або "Василькові 10 років". На Пролозі це запишеться у виді

'Сьогодні сонячно'.

'Василькові 10 років'.

Якщо ви збережете ці факти у файлі і потім завантажите його, то можна ставити питання інтерпретатору Прологу (нагадаємо, що запит вводиться після запрошення Прологу, яке у більшості версій має вигляд ?-), наприклад,

```
?- 'Сьогодні сонячно'.
Yes

?- 'Василькові 10 років'.
Yes

?- 'Сьогодні сонячно', 'Василькові 10 років'.
Yes
```

Кома між фактами в останньому запиті означає операцію логічного **і** (кон'юнкцію).

Така форма запису відповідає логіці висловлювань, можливості якої, як вже говорилося, досить обмежені. Ми не можемо поставити, наприклад, питання про те, скільки років Василькові. Набагато зручніше використати факти, що *параметризуються*, роботу з якими підтримує логіка предикатів. На Пролозі факт може бути записаний у вигляді предиката, аргументи якого є символьними або числовими константами.

Правила. Пролог може містити правила, що дозволяють отримувати додаткові знання про той світ, який описує програма. Правило задає новий предикат через визначені раніше. Правило складається з голови (предиката) і тіла (послідовності предикатів, розділених комами). Голова і тіло розділені знаком **:–** і, подібно до кожної фрази Прологу, правило повинне закінчуватися крапкою. Кома в тілі правила означає кон'юнкцію (&& логічне І).

Знак **:–** є схематичний запис стрілки **(<-)** і показує, що з правої частини виходить ліва. Цей знак читається як "*якщо"*. Інтуїтивний сенс правила полягає в тому, що мета, що є головою, буде істинною, якщо Пролог зможе показати, що усі вирази (підцілі) в тілі правила є істинними.

Приклад

Правило, яке визначає відношення дитина/2 через відношення батько/2, запишеться таким чином:

дитина(X, Y) :- батько(Y, X).

Це означає, що якщо людина Y є для людини X батьком, то X є дитиною Y. Тут X і Y - змінні. Нагадаємо, що запис дитина/2 показує, що предикат дитина є функцією від двох аргументів.

d. Предикати

Предикат - це логічна функція від одного або декількох аргументів, тобто функція, яка дає множину з двох значень : істина і брехня. Предикат Прологу записується у вигляді складеного терма:

ім'я_предиката(аргументи).

Аргументи перераховуються через кому і є якимись об'єктами або властивостями об'єктів, а ім'я предиката означає зв'язок або відношення між аргументами. Предикат однозначно визначається парою: ім'я і кількість аргументів. Два предикати з однаковим ім'ям, але різною кількістю аргументів, вважаються різними. Кількість параметрів предиката називається його *арністю* (arity). При описі предиката арність вказують після його імені, розділяючи їх символом '/' (слеш).

Як правило, імена предикатів і аргументів записуються в називному відмінку. Пропуски в них не допускаються, тому в якості роздільників в символьних константах використовується символ підкреслення.

Приклад:

Факт "Коля працює слюсарем" на Пролозі запишеться таким чином:

професія(Коля, слюсар).

Тут предикат професія/2 має два аргументи: перший означає ім'я людини, а другий - професію. Факт "Борису 10 років" можна представити у виді:

вік("Борис", 10).

Порядок аргументів предиката пов'язаний з сенсом факту і тому не змінюваний. При записі фактів потрібно пам'ятати, що:

ім'я факту розпочинається з великої букви;

запис кожного факту закінчується крапкою.

У наведених вище прикладах професія/2 і вік/2 - предикати (складені терми), Коля і слюсар - атоми, 10 - число, "Борис" - рядок. Детальніше про види термів Прологу розповідається в наступному розділі.

База даних на Пролозі - це сукупність фактів. В процесі роботи у базу даних можна додавати нові факти, видаляти або змінювати старі.

Приклад

Складемо базу даних з наступних фактів: "слон більший, ніж кінь", "кінь більший, ніж осел", "осел більший, ніж собака" і "осел більший, ніж мавпа" :

більше(слон, кінь).
більше(кінь, осел).
більше(осел, собака).
більше(осел, мавпа).

Ми використали предикат більше/2, що має два параметри.

Збережемо цю базу даних в текстовому файлі і потім познайомимо Пролог з нею. Тепер можна формулювати запити до інтерпретатора Прологу :

?- більше(слон, кінь).
Yes

?- більше(кінь, слон).
No

3. Завдання на лабораторну роботу

В ході цієї роботи необхідно:

1. Побудувати базу знань з будь-якої предметної області. (не менше 10 фактів, трьох правил)

2. Скласти набір запитів до бази знань з п.1. (не менше 5-ти запитів з одною відповіддю, та не менше 3-х запитів із множинними відповідями, у запитах мають брати участь усі правила)

Приклади предметних областей :

1. Відношення підлеглості в групі проектувальників.
2. Відношення зв'язків в комплексній програмі, що розробляється групою програмістів.
3. Ієрархічні відношення в організаційній моделі ВНЗ.
4. Відношення, необхідні для нарахування стипендії.
5. Відношення, що перевіряються при видачі диплому з відмінністю.
6. Відношення, контрольовані для видачі напряму в аспірантуру.
7. Відношення між курсами по навчальних планах.
8. Відношення в організаційній ієрархії гуртожитку.

4. Приклад виконання

Лістинг програми :

```
% Будуємо базу знань
% Факти, представлені в процедурі student
% встановлюють відповідність студентів і їх середніх балів
% student(Name, SrBal)
student('Ivanenko', 4.5).
student('Vasulenko', 3.0).
student('Vitrenko', 4.3).
student('Petrenko', 3.2).
student('Levchenko', 4.6).
student('Sidorenko', 5.0).

% Складаємо правило "Стипендія"
% Name - ім'я студента
% Результат виклику цього правила :
% 1. якщо параметром передати конкретне ім'я Prolog -система дасть відповідь
%    true якщо студент має стипендію або fail якщо не має.
```

```
% 2. якщо параметром передати змінну, то буде виведений список студентів
%    які мають стипендію
stipendiya(Name) :- student(Name, SrBal), SrBal > 4.

% Визначимо ще одну пролог-процедуру company
% Ця процедура визначатиме мінімальний середній бал
% необхідний студентові для роботи в цій компанії
% company(Name, MinSrBal)
company('Microsoft', 5.0).
company('Apple', 4.9).
company('IBM', 4.5).
company('IT-west', 4.0).
company('Vasya and Co',3.0).

% Складемо правило чи "може студент працювати в компанії"
% Тут теж можна вводити конкретні значення для відповіді "да-нет"
% або змінні для отримання списку компаній або студентів.
isAbleToWork(NameStudent, NameCompany) :-
                    student(NameStudent,SrBalStudent),
                    company(NameCompany,SrBalCompany),
                    SrBalStudent>=SrBalCompany.
```

Запити:

```
?- stipendiya('Levchenko').
true.

?- stipendiya('Vasulenko').
false.

?- stipendiya(S).
S = 'Ivanenko' ;
S = 'Vitrenko' ;
S = 'Levchenko' ;
S = 'Sidorenko'.

?- company('Microsoft',X).
X = 5.0.

?- isAbleToWork('Levchenko','Microsoft').
false.

?- isAbleToWork('Sidorenko','Microsoft').
true.

?- isAbleToWork('Levchenko',X).
X = 'IBM' ;
X = 'IT-west' ;
X = 'Vasya and Co'.

?- isAbleToWork(X,'Microsoft').
X = 'Sidorenko'.

?- isAbleToWork(X,'IBM').
X = 'Ivanenko' ;
X = 'Levchenko' ;
```

X = 'Sidorenko'.

?- isAbleToWork(X,Y).
X = 'Ivanenko',
Y = 'IBM' ;
X = 'Ivanenko',
Y = 'IT-west' ;
X = 'Ivanenko',
Y = 'Vasya and Co' ;
X = 'Vasulenko',
Y = 'Vasya and Co' ;
X = 'Vitrenko',
Y = 'IT-west' ;
X = 'Vitrenko',
Y = 'Vasya and Co' ;
X = 'Petrenko',
Y = 'Vasya and Co' ;
X = 'Levchenko',
Y = 'IBM' ;
X = 'Levchenko',
Y = 'IT-west' ;
X = 'Levchenko',
Y = 'Vasya and Co' ;
X = 'Sidorenko',
Y = 'Microsoft' ;
X = 'Sidorenko',
Y = 'Apple' ;
X = 'Sidorenko',
Y = 'IBM' ;
X = 'Sidorenko',
Y = 'IT-west' ;
X = 'Sidorenko',
Y = 'Vasya and Co'.