**Лабораторна робота №4**

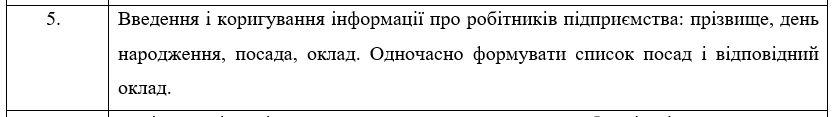
**Введення-виведення і робота з базою**

**Цілі роботи:**

* ознайомлення з предикатами введення-виведення у мові Prolog;
* вивчення та одержання практичних навичок з формуванням та змінні бази даних програми.

**Завдання:**

Завантажити з файлу список фактів, виконати дії відповідно завдання, а по закінченні сеансу роботи зберегти стан бази даних в файл.



Короткі теоретичні відомості

***Предикати введення-виведення***

Prolog-програма може читати дані з вхідного потоку і записувати в вихідний потік. У якості вхідного та вихідного потоку може бути файл або термінал (клавіатура і монітор відповідно). Дані, які введені з клавіатури і дані, які виведені на монітор, розглядаються як потік, якій має назву user. У будь який час існує один поточний вхідний і один вихідний потік. За замовчування поточним потоком є термінал, однак його можна змінити на файл, а після закінчення роботи з файлом відновити потік user.

Для зміни вхідного потоку використовується предикат

see(<імя файлу>)

Якщо вказаний файл існує, ціль see успішна, поточним вхідним потоком становиться файл і всі наступні операції читання здійснюються з файлу. Для відновлення потоку user необхідно застосувати предикат see(user) або предикат seen без параметрів.

***Предикати для роботи з базою даних програми***

У процесі виконання програми зміст бази даних може бути змінений. Для цього існують предикати (див. табл. 8.3) , які дозволяють додавати або вилучати твердження із бази даних.

|  |  |
| --- | --- |
| Таблиця 8.3 – Предикати для роботи з базою даних | |
| **Назва** | **Призначення** |
| assert(Х)  assertz(Х) | розміщує нове твердження X в кінці бази даних, або додає його до подібних за арністю та функтором тверджень за умови їх присутності |
| asserta(Х) | розміщує нове твердження X у початок бази даних |
| retract(Х) | вилучає задане твердження X із бази даних |
| abolish(Р/А) | вилучає всі подібні твердження з заданими функтором P і арністю A |

**Методичні вказівки**

***Особливості роботи з файлами***

Використання будь-якого предикату читання з файлу припускає, що файл існує. Щоб запобігти виникненню помилки у разі, якщо файл не існує, слід застосувати предикат exists\_file(F), де F – ім’я файлу. Предикат є вдалим, якщо файл F існує.

При читанні даних з файлу іноді необхідна перевірка кінця файлу. У мові Prolog існує атом end\_of\_file, що дає можливість виконати перевірку кінця файлу після кожного читання. Фрагмент програми, представлений нижче, демонструє подібну перевірку:

p :- read(X), checkread(X).

checkread(end\_of\_file) :- write(‘Find end of file.’), !.

checkread(X) :- <Обробка терму **X**>.

***Перевірка конкретизації змінної***

У програмі іноді важливо визначити стан конкретизації змінної виконання подальших дій. Предикат var(X) є вдалим, якщо X – неконкретизована змінна. Предикат nonvar(X) є вдалим, якщо X – конкретизована змінна. Приклад застосування цих предикатів представлений нижче.

**Рішення**

1. Аналіз
2. Текст програми

:-dynamic employee/5, position/3.

guide :- write('\* \* \* Prolog DB (v0.0.1) \* \* \*'), nl, nl,

   write('Available predicates:'), nl,

   write('?- empl(Id)'), nl,

   write('?- emplAll()'), nl,

   write('?- emplAdd(Id, PosId, FN, LN, Birthday)'), nl,

   write('?- emplEdit(Id, PosId, FM, LN, Birthday)'), nl,

   write('?- emplDel(Id)'), nl,

   write('?- pos(Id)'), nl,

   write('?- posAll()'), nl,

   write('?- posAdd(Id, Name, Salary) '), nl,

   write('?- posEdit(Id, Name, Salary)'), nl,

   write('?- posDel(Id)'), nl,

   write('?- guide()'), nl,

   write('?- save()'), nl.

% Run, Read, Save

run :- write('Loading data... '),

   (exists\_file('db.txt'); write('Can\'t load data. DB file not found'), nl, fail),

   see('db.txt'), read(X), readNext(X), seen(),

   nl, nl, guide, !.

readNext(end\_of\_file) :- done, nl, !.

readNext(X) :- assert(X), read(Y), readNext(Y).

save :- write('Saving... '),

   (exists\_file('db.txt'); write('Can\'t save data. DB file not found'), nl, fail),

   tell('db.txt'),

   findall(EmplId, employee(EmplId, \_, \_, \_, \_), EmplL), saveEmployees(EmplL),

   findall(PosId, position(PosId, \_, \_), PosL), savePositions(PosL),

   told(), done, nl, !.

saveEmployees(EmployeeIds) :- maplist(saveEmployee, EmployeeIds).

saveEmployee(Id) :- employee(Id, PosId, FN, LN, Birth),

   writeq(employee(Id, PosId, FN, LN, Birth)), write('.'), nl.

savePositions(PositionIds) :- maplist(savePosition, PositionIds).

savePosition(Id) :- position(Id, Name, Salary),

   writeq(position(Id, Name, Salary)), write('.'), nl.

% Employee

empl(Id) :- employee(Id, PosId, FN, LN, Birth),

   position(PosId, PosName, Salary),

   format('| ~a~t~5+ | ~a ~a~t~20+ | ~a~t~15+ | ~a~t~20+ | ~a~t~10+ |~n', [Id, FN, LN, Birth, PosName, Salary]).

emplDel(Id) :- (employee(Id, \_, \_, \_, \_); write('Employee not found')),

   retract(employee(Id, \_, \_, \_, \_)), !.

emplAdd(Id, PosId, FN, LN, Birth) :- (not(employee(Id, \_, \_, \_, \_)); write('Employee already exist. Use ?- emplEdit() instead'), nl, fail),

   (position(PosId, \_, \_); write('Position not found'), nl, fail),

   assert(employee(Id, PosId, FN, LN, Birth)),

   done, !.

emplEdit(Id, PosId, FN, LN, Birth) :- (employee(Id, \_, \_, \_, \_); write('Employee not found'), nl, fail),

   (position(PosId, \_, \_); write('Position not found'), nl, fail),

   emplDel(Id),

   assert(employee(Id, PosId, FN, LN, Birth)),

   done, !.

emplLine() :- format('|~`-t~71||~n').

emplAll() :- emplLine, emplHead, emplLine, empl(\_), emplLine, fail.

emplHead() :- format('| ~a~t~5+ | ~a~t~20+ | ~a~t~15+ | ~a~t~20+ | ~a~t~10+ |~n', ['Id', 'Name', 'Birthday', 'Position', 'Salary']).

% Position

pos(Id) :- position(Id, Name, Salary),

   format('| ~a~t~5+ | ~a~t~20+ | ~a~t~10+ |~n', [Id, Name, Salary]).

posDel(Id) :- (position(Id, \_, \_); write('Position not found'), fail),

   (not(employee(\_, Id, \_, \_, \_)); write('There are some employees on that position. Can\'t remove it.'), fail),

   retract(position(Id, \_, \_)), !.

posAdd(Id, Name, Salary) :- (not(position(Id, \_, \_)); write('Position already exist. Use ?- posEdit() instead'), nl, fail),

   assert(position(Id, Name, Salary)),

   done, !.

posEdit(Id, Name, Salary) :- ((position(Id, \_, \_); write('Position not found'), nl, fail)),

   retract(position(Id, \_, \_)),

   assert(position(Id, Name, Salary)),

   done, !.

posAll() :- posLine, posHead, posLine, pos(\_), posLine, fail.

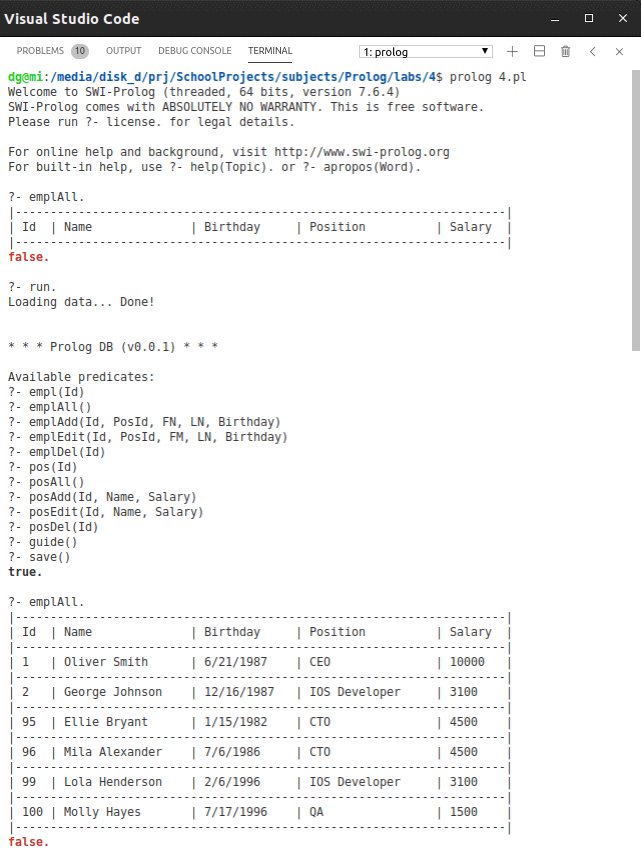
posHead() :- format('| ~a~t~5+ | ~a~t~20+ | ~a~t~10+ |~n', ['Id', 'Name', 'Salary']).

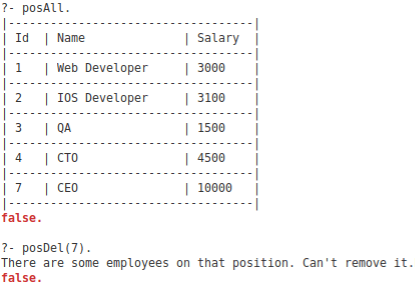
posLine() :- format('|~`-t~36||~n').

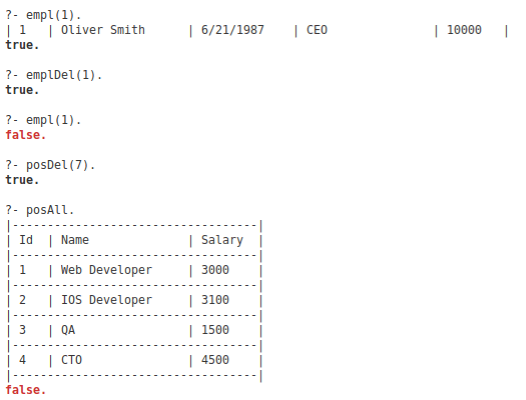
% Other

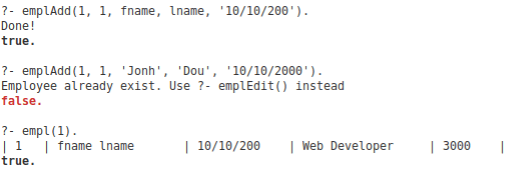
done  :- write('Done!').

1. Результат виконання програми

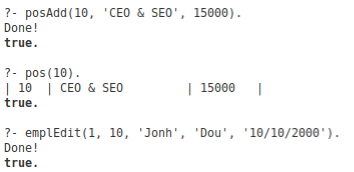


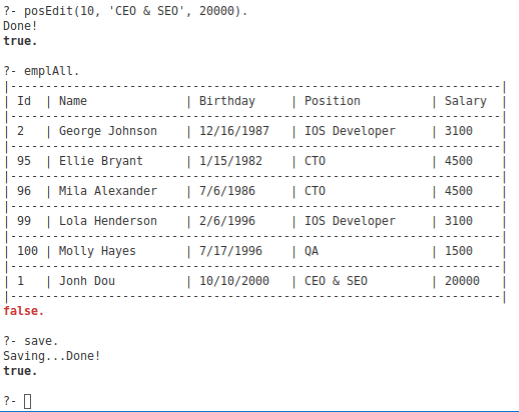








****

****

**Висновок:** на цій лабораторній роботі я ознайомився з предикатами введення-виведення у мові Prolog; вивчив та одержав практичні навички з формування та змінною бази даних програми.