Міністерство освіти і науки України
Національний технічний університет України „КПІ”
Факультет інформатики та обчислювальної техніки

Кафедра автоматизованих систем обробки
інформації та управління

ЗВІТ

до лабораторної роботи № 8
з предмету:

„МУЛЬТИПАРАДИГМЕННЕ ПРОГРАМУВАННЯ”

Виконав студент *ІП-61 Кушка Михайло Олександрович, 3-й курс, ІП-6116*

(№ групи, прізвище, ім’я, по батькові, курс, номер залікової книжки)

Прийняв *Очеретяний О. К.*

(посада, прізвище, ім’я, по батькові)

Київ 2018

ЗМІСТ

1. ПОСТАНОВКА ЗАДАЧІ

Завдання:

1. Проінсталювати на власному комп'ютері систему програмування GNUPROLOG та систему редагування текстів программ SciTE (Science Text Editor).

2. Скласти на мові Prolog дерево родинних відношень, використовуючи предикат **roditel** з двома параметрами: ім'я одного з батьків та ім'я дитини. Написати на мові Prolog та запустити наступні запити:

•“Хто є і батьками, і має батьків”

•“Хто не має дітей”

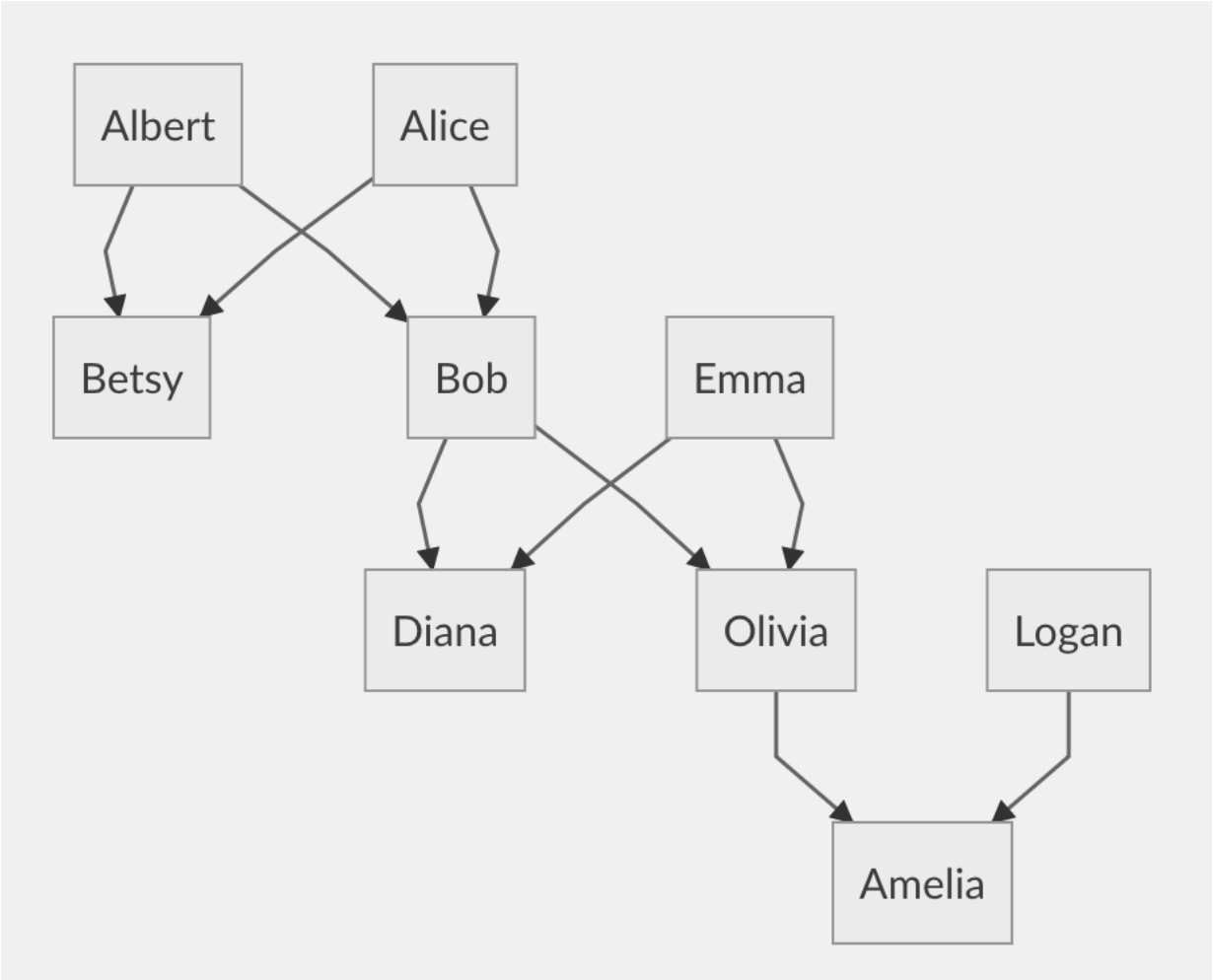

Рисунок 1 - Схема родинних зв'язків

3. Програму доповнити новими фактами, що дозволяють побудувати правила для визначення наступних цілей-предикатів:

- батько
- мати
- син
- дочка
- брат
- сестра
- дядько
- тітка
- дід
- баба
- онук
- онучка
- небіж
- небога
- одружені
- теща
- тесть
- свекруха
- свекор
- зять
- невістка
- свояк
- своячка
- дівер
- внучатий небіж
- внучата небога

2. РЕЗУЛЬТАТИ РОБОТИ

```
| ?- aunt(olivia, X).

X = betsy ? ;

no
| ?- mother_in_law(logan, X).

X = emma ? ;

no
| ?- grand_nephew(alice, X).

no
| ?- grand_nephew(emma, X).

no
| ?- grand_nephew(betsy, X).

no
| ?- great_niece(betsy, X).

X = amelia ? ;

no
| ?- part(bob, X).

X = emma ? ;

X = emma ? ;

no
| ?- part(emma, X).

X = bob ? ;

X = bob ? ;

no
| ?- halt.
```

Рисунок 2 – приклад роботи деяких запитів

3. ВИСНОВОК

В ході даної лабораторної роботи я познайомився з мовою програмування Prolog, ознайомився з її базовим синтаксисом та навчився писати прості програми.

4. КОД ПРОГРАМИ

```
male(albert).
male(bob).
male(logan).

female(alice).
female(betsy).
female(emma).
female(diana).
female(olivia).
female(amelia).

parent(albert, betsy).
parent(albert, bob).
parent(alice, betsy).
parent(alice, bob).
parent(bob, diana).
parent(bob, olivia).
parent(emma, diana).
parent(emma, olivia).
parent(olivia, amelia).
parent(logan, amelia).

% has parent & is a parent
both :-
    parent(_, X),
    parent(X, _),
    format('~w has parents and is a parent ~n', [X]).


who_has_no_children :-
    parent(X, Y),
    no_children(Y),
    format('~w has no children ~n', [Y]).

no_children(Name) :-
    \+ parent(Name, X).

father(Name, X) :-
    parent(X, Name),
    male(X).

mother(Name, X) :-
    parent(X, Name),
    female(X).

son(Name, X) :-
    parent(Name, X),
    male(X).

daughter(Name, X) :-
    parent(Name, X),
    female(X).

brother(Name, X) :-
    father(Name, F),
    son(F, X),
```

```
    \+ X=Name.

sister(Name, X) :-
    father(Name, F),
    daughter(F, X),
    \+ X=Name.

uncle(Name, X) :-
    mother(Name, M),
    brother(M, X).

uncle(Name, X) :-
    father(Name, F),
    brother(F, X).

aunt(Name, X) :-
    mother(Name, M),
    sister(M, X).

aunt(Name, X) :-
    father(Name, F),
    sister(F, X).

grandpa(Name, X) :-
    father(Name, F),
    father(F, X).

grandpa(Name, X) :-
    mother(Name, M),
    father(M, X).

grandma(Name, X) :-
    father(Name, F),
    mother(F, X).

grandma(Name, X) :-
    mother(Name, M),
    mother(M, X).

grandson(Name, X) :-
    son(Name, S),
    son(S, X).

grandson(Name, X) :-
    daughter(Name, D),
    son(D, X).

granddaughter(Name, X) :-
    son(Name, S),
    daughter(S, X).

granddaughter(Name, X) :-
    daughter(Name, D),
    daughter(D, X).

nephew(Name, X) :-
    brother(Name, B),
    son(B, X).
```

```
nephew(Name, X) :-
    sister(Name, S),
    son(S, X).

niece(Name, X) :-
    brother(Name, B),
    daughter(B, X).

niece(Name, X) :-
    sister(Name, S),
    daughter(S, X).

part(Name, X) :-
    son(Name, S),
    mother(S, X),
    \+ X=Name.

part(Name, X) :-
    daughter(Name, D),
    mother(D, X),
    \+ X=Name.

part(Name, X) :-
    son(Name, S),
    father(S, X),
    \+ X=Name.

part(Name, X) :-
    daughter(Name, S),
    father(S, X),
    \+ X=Name.

mother_in_law(Name, X) :-
    part(Name, W),
    mother(W, X).

father_in_law(Name, X) :-
    part(Name, W),
    father(W, X).

son_in_law(Name, X) :-
    daughter(Name, D),
    part(D, X).

daughter_in_law(Name, X) :-
    son(Name, S),
    part(S, X).

brother_in_law(Name, X) :-
    female(Name),
    sister(Name, S),
    part(S, X).

sister_in_law(Name, X) :-
    male(Name),
    part(Name, W),
    sister(W, X).
```

```
diver(Name, X) :-
    female(Name),
    part(Name, H),
    brother(H, X).

grand_nephew(Name, X) :-
    brother(Name, B),
    grandson(B, X).

grand_nephew(Name, X) :-
    sister(Name, S),
    grandson(S, X).

great_niece(Name, X) :-
    brother(Name, B),
    granddaughter(B, X).

great_niece(Name, X) :-
    sister(Name, S),
    granddaughter(S, X).
```