МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования «Тульский государственный университет»

Институт прикладной математики и компьютерных наук

Отчет по лабораторной работе №1

«Знакомство с Visual Prolog 8»

по дисциплине «Функциональное и логическое программирование»

Выполнил студент группы 220681

Шайхаттаров Д.В.

Проверил:

асс. Савенков П. А.

Тула 2020

# **ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ**

Узнать и усвоить базовые понятия языка и основы написания программ в среде Visual Prolog на языке Пролог

# **ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ**

Создайте предикат, проверяющий, являются ли два человека

* сестрами;
* братьями;
* бабушкой и внуком (внучкой);
* дядей и племянником (племянницей);
* двоюродными братьями;
* родственниками.

# **ОПИСАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ**

Предметной областью приложения является семья, где заранее известны имена, пол и связь ребёнок-родитель. Схема описание предметной области в формате IDEF5 предоставлена на рисунке 1.

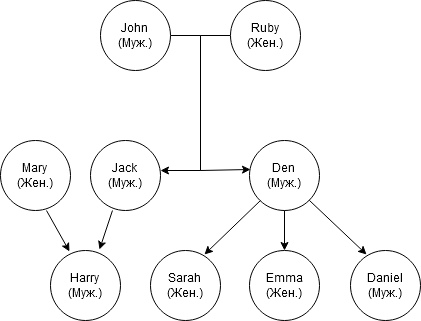


Рисунок 1 – Схема ПрО в формате IDEF5

# **ТЕСТИРОВАНИЕ**

При запуске программы и вводе значений «Ruby» и «Emma» в консоль выводятся следующие результаты (Рисунок 2).

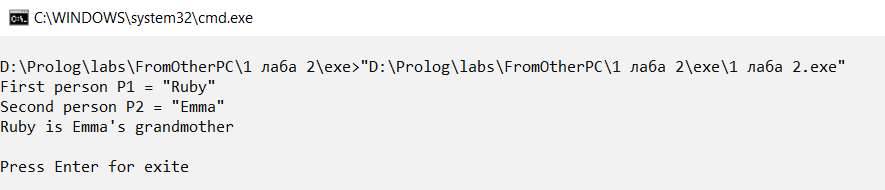


Рисунок 2 – Результаты тестирования

# **ЗАКЛЮЧЕНИЕ**

В ходе данной лабораторной работы были изучены и усвоены базовые понятия языка и основы написания программ в среде Visual Prolog на языке Пролог.

**Приложение**

**Исходный код программы задания**

implement main  
    open core, console  
  
domains  
    gender = female; male.  
  
class facts  
    person : (string Name, gender Gender).  
    parent : (string Person, string Parent).  
  
clauses  
    person("Sarah", female()).  
    person("Emma", female()).  
    person("Daniel", male()).  
    person("Den", male()).  
    person("Harry", male()).  
    person("Jack", male()).  
    person("Ruby", female()).  
    person("Mary", female()).  
    person("John", male()).  
    parent("Sarah", "Den").  
    parent("Emma", "Den").  
    parent("Daniel", "Den").  
    parent("Jack", "Ruby").  
    parent("Den", "Ruby").  
    parent("Harry", "Jack").  
    parent("Jack", "John").  
    parent("Den", "John").  
    parent("Harry", "Mary").  
  
class predicates  
    siblings : (string Person1, string Person2) nondeterm.  
clauses  
    siblings(Person1, Person2) :-  
        parent(Person1, Parent),  
        parent(Person2, Parent).  
  
class predicates  
    sisters : (string Sis1, string Sis2) nondeterm.  
clauses  
    sisters(Sis1, Sis2) :-  
        siblings(Sis1, Sis2),  
        person(Sis1, female()),  
        person(Sis2, female()).  
  
class predicates  
    brothers : (string Bro1, string Bro2) nondeterm.  
clauses  
    brothers(Bro1, Bro2) :-  
        siblings(Bro1, Bro2),  
        person(Bro1, male()),  
        person(Bro2, male()).  
  
class predicates  
    grandmother : (string Person, string Grandmother) nondeterm.  
clauses  
    grandmother(Person, Grandmother) :-  
        parent(Person, Parent),  
        parent(Parent, Grandmother),  
        person(Grandmother, female()).  
  
class predicates  
    uncle : (string Person, string Uncle) nondeterm.  
clauses  
    uncle(Person, Uncle) :-  
        parent(Person, Parent),  
        siblings(Parent, Uncle),  
        person(Uncle, male()).  
  
class predicates  
    cousinsMen : (string Cousin1, string Cousin2) nondeterm.  
clauses  
    cousinsMen(Cousin1, Cousin2) :-  
        parent(Cousin1, Parent1),  
        parent(Cousin2, Parent2),  
        siblings(Parent1, Parent2),  
        person(Cousin1, male()),  
        person(Cousin2, male()).  
  
class predicates  
    relatives : (string Person1, string Person2, integer N) determ.  
clauses  
    relatives(Person1, Person2, N) :-  
        N < 7,  
        parent(Person1, P),  
        relatives(P, Person2, N + 1),  
        !.  
    relatives(Person1, Person2, N) :-  
        N < 7,  
        parent(P, Person1),  
        relatives(P, Person2, N + 1),  
        !.  
    relatives(Person1, Person1, \_N) :-  
        !.  
  
class predicates  
    findRelatives : (string Person1, string Person2) nondeterm.  
clauses

    findRelatives(Person, Person) :-  
        !.  
    findRelatives(Person1, Person2) :-  
        brothers(Person1, Person2),  
        stdIO::writef("% and % are brothers\n", Person1, Person2)  
        or  
        sisters(Person1, Person2),  
        stdIO::writef("% and % are sisters\n", Person1, Person2)  
        or  
        grandmother(Person1, Person2),  
        stdIO::writef("% is %'s grandmother\n", Person2, Person1)  
        or  
        grandmother(Person2, Person1),  
        stdIO::writef("% is %'s grandmother\n", Person1, Person2)  
        or  
        uncle(Person1, Person2),  
        stdIO::writef("% is %'s uncle\n", Person2, Person1)  
        or  
        uncle(Person2, Person1),  
        stdIO::writef("% is %'s uncle\n", Person1, Person2)  
        or  
        cousinsMen(Person1, Person2),  
        stdIO::writef("% is %'s cousin and they are men\n", Person1, Person2)  
        or  
        relatives(Person1, Person2, 0),  
        stdIO::writef("% and % are relatives\n", Person1, Person2).  
  
clauses  
    run() :-  
        init(),  
        write("First person P1 = "),  
        P1 = read(),  
        write("Second person P2 = "),  
        P2 = read(),  
        if not(findRelatives(P1, P2)) or P1 = P2 then  
            write("They are not relatives"),

            nl  
        end if,  
        clearInput(),  
        write("Press Enter for exit"),  
        \_ = readLine().  
  
end implement main  
  
goal  
    console::runUtf8(main::run).