Федеральное государственное бюджетное образовательное учреждение высшего образования
Национальный исследовательский университет «Московский энергетический институт»
Кафедра ВМСС

Лабораторная работа №9

«Разработка приложения SWI-пролог и C#»

Курс: Технологии разработки программного обеспечения

Группа: А-07м-23

Выполнил: Кретов Н.В.

Проверила: Раскатова М.В.

Москва 2024

СОДЕРЖАНИЕ

1. Цель

Главной задачей данной лабораторной работы является ознакомление с языком логического программирования Prolog и разработки интерфейса взаимодействия пользователя с разработанной базой знаний на языке высокого уровня C#.

2. Задание

На языке SWI-Prolog разработать дерево семейных отношений (глубина – не менее 7). Записать представление разработанного дерева на языке SWI-пролог. Реализовать 5-6 различных правил, включая рекурсивные правила. Задать системе различные вопросы.

На языке SWI-Prolog разработать семантическую сеть, представленную различными фактами, «ширина» - 2-3, «глубина» - 7-8. Предметную область выбрать самостоятельно. Например: транспорт, спорт, животные, спортивные игры, музыка, самолеты. Разработать не менее 7 различных правил (на различных уровнях сети). На основании информации, явно заданной в сети, вывести другие факты, используя механизм наследования классами свойств суперклассов. Система должна сформировать ответы на вопросы, которые явно не содержатся в базе фактов системы, выполнив логический вывод с помощью механизма наследования.

На ЯП C# разработать инструментальную часть + графический интерфейс.

3. Описание предметной области

В качестве семейного древа было выбрано семейное древо династии Романовых (глубина 10 поколений, ширина – 9).

В качестве предметной области для семантической сети была выбрана иерархия предметов в игре жанра RPG.

Для реализации инструментария и графического интерфейса был выбран ЯП C#. Планируется использования библиотеки CSProlog, позволяющей использовать язык SWI-Prolog при помощи библиотечных функций. Графический интерфейс планируется разработать при помощи WinForms.

4. Разработка

Для семейного древа были разработаны следующие факты и правила:

- parent(X, Y). - Факт родительства X по отношению к Y.
- woman(X). - Факт принадлежности X к женскому полу.
- man(X). - Факт принадлежности X к мужскому полу.
- married(X, Y). - Факт брака между X и Y.
- check_married(X, Y) :- married(X, Y), married(Y, X). - Правило проверки факта брака между X и Y.
- father(X, Y) :- parent(X, Y), man(X). - Правило проверки, является ли X отцом для Y.
- mother(X, Y) :- parent(X, Y), woman(X). - Правило проверки, является ли X матерью для Y.
- brother(X, Y) :- parent(Z, X), parent(Z, Y), man(X), X \= Y. – Правило проверки, является ли X братом для Y.
- sister(X, Y) :- parent(Z, X), parent(Z, Y), woman(X), X \= Y. – Правило проверки, является ли X сестрой для Y.
- son(X, Y) :- parent(Y, X), man(X). - Правило проверки, является ли X братом для Y.
- daughter(X, Y) :- parent(Y, X), woman(X). - Правило проверки, является ли X сестрой для Y.
- husband(X, Y) :- check_married(X, Y), man(X). - Правило проверки, является ли X мужем для Y.
- wife(X, Y) :- check_married(X, Y), woman(X). - Правило проверки, является ли X женой для Y.
- grandfather(X, Y) :- parent(X, Z), parent(Z, X), man(X). – Правило проверки, является ли X дедом для Y.
- grandmother(X, Y) :- parent(X, Z), parent(Z, X), woman(X). – Правило проверки, является ли X бабушкой для Y.
- grandson(X, Y) :- parent(Z, Y), parent(X, Z), man(Y). - Правило проверки, является ли Y внуком для X.

- granddaughter(X, Y) :- parent(Z, Y), parent(X, Z), woman(Y). – Правило проверки, является ли Y внучкой для X.
- predecessor(X, Y) :- parent(X, Y). predecessor(X, Y) :- parent(X, Z),
- predecessor(Z, Y). - Рекурсивное правило проверки, является ли X предком для Y.

Для семантической сети были разработаны следующие факты и правила:

- connect (X, Y) - Факт принадлежности X к классу Y.
- property (X, Y) - Факт свойства Y у X.
- Inherit :- connect(X, Y), property_too(Y, Z). - Правило наследования принадлежности классу.
- property_too :- property(X, Z); inherit(X, Z). - Правило наследования свойств.
- descendant :- connect(X, Y). descendant :- connect(X, Z), descendant(Z, Y). - Рекурсивное правило проверки, является ли X наследным от Y.
- check_property :- property_too(X, Z). - Правило проверки свойства Z у X.
- properties :- findall(Property, check_property(X, Property), Properties). - Правило поиска всех свойств X.
- descendants :- findall(Descendant, descendant(Descendant, Y), Descendants). - - Правило поиска все наследных классов для Y.

5. Дерево семейных отношений

На рис. 1 представлено дерево семейных отношений.

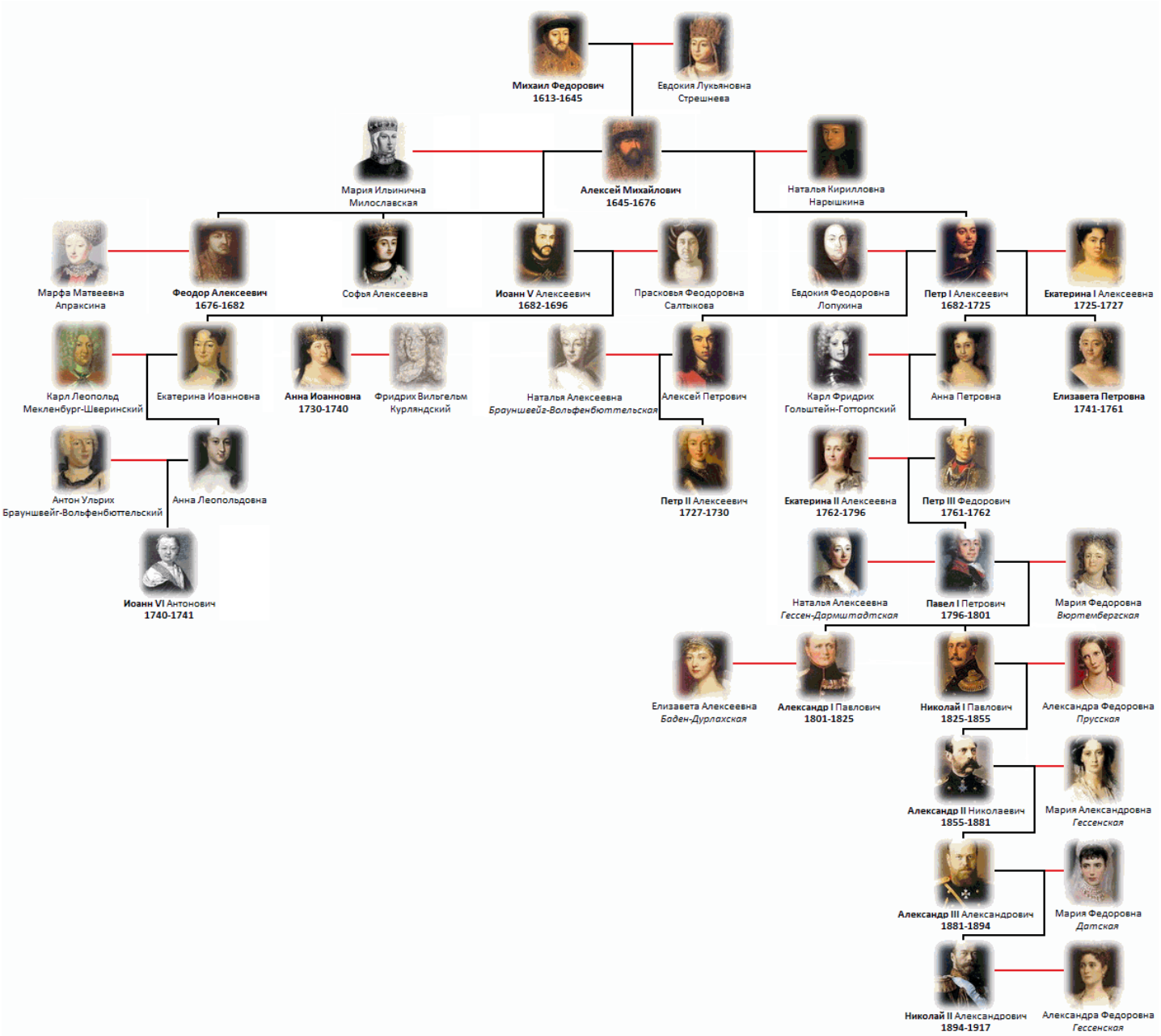


Рис. 1. Дерево семейных отношений (династия Романовых)

С листингом программы на Prolog, выполняющей задание 1, можно ознакомиться в приложении А.

6. Семантическая сеть

На рис. 2 представлена семантическая сеть предметов из игры жанра RPG

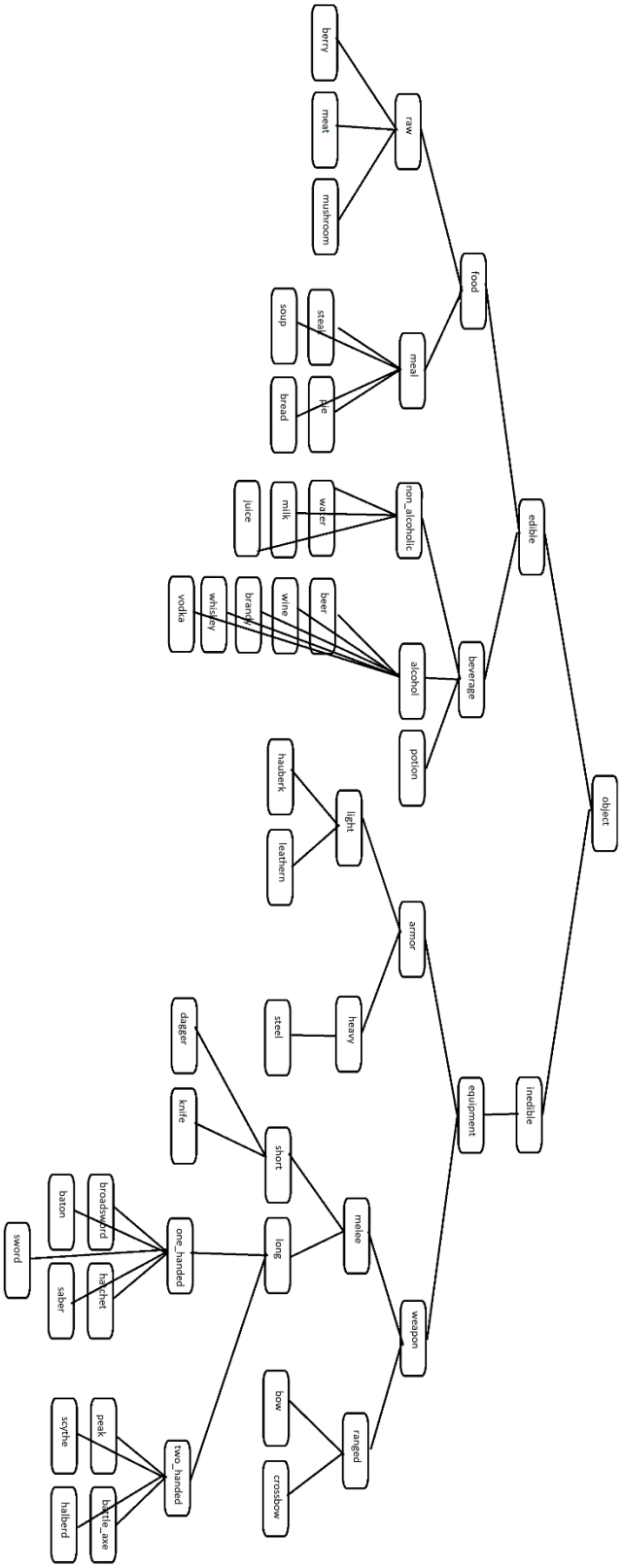


Рис. 2. Семантическая сеть (предметы из игры жанра RPG)

С листингом программы на Prolog, выполняющей задание 2, можно ознакомиться в приложении Б.

7. Результат работы программы

На рис. 3-10 представлены результаты работы программы.

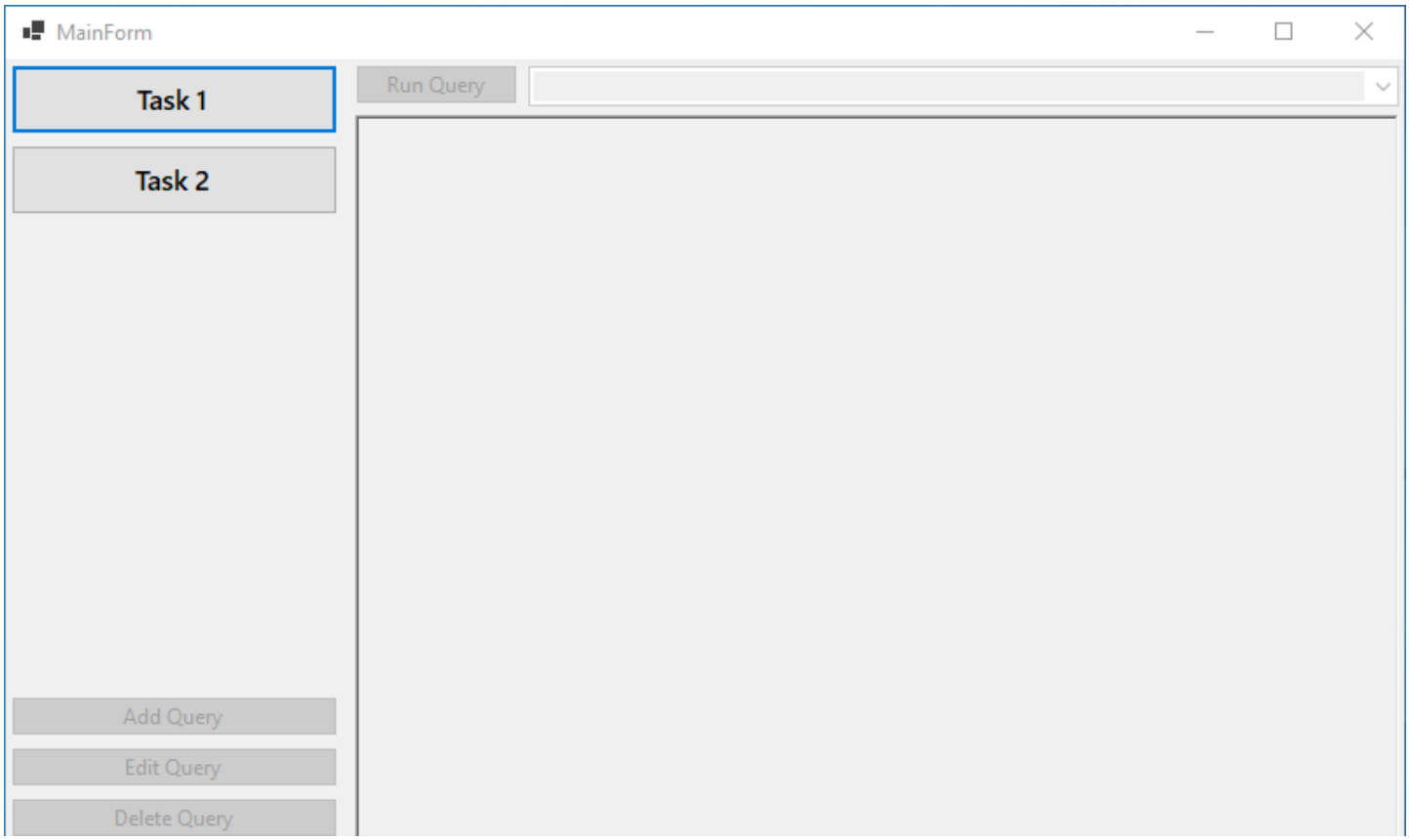


Рис. 3. Запуск программы

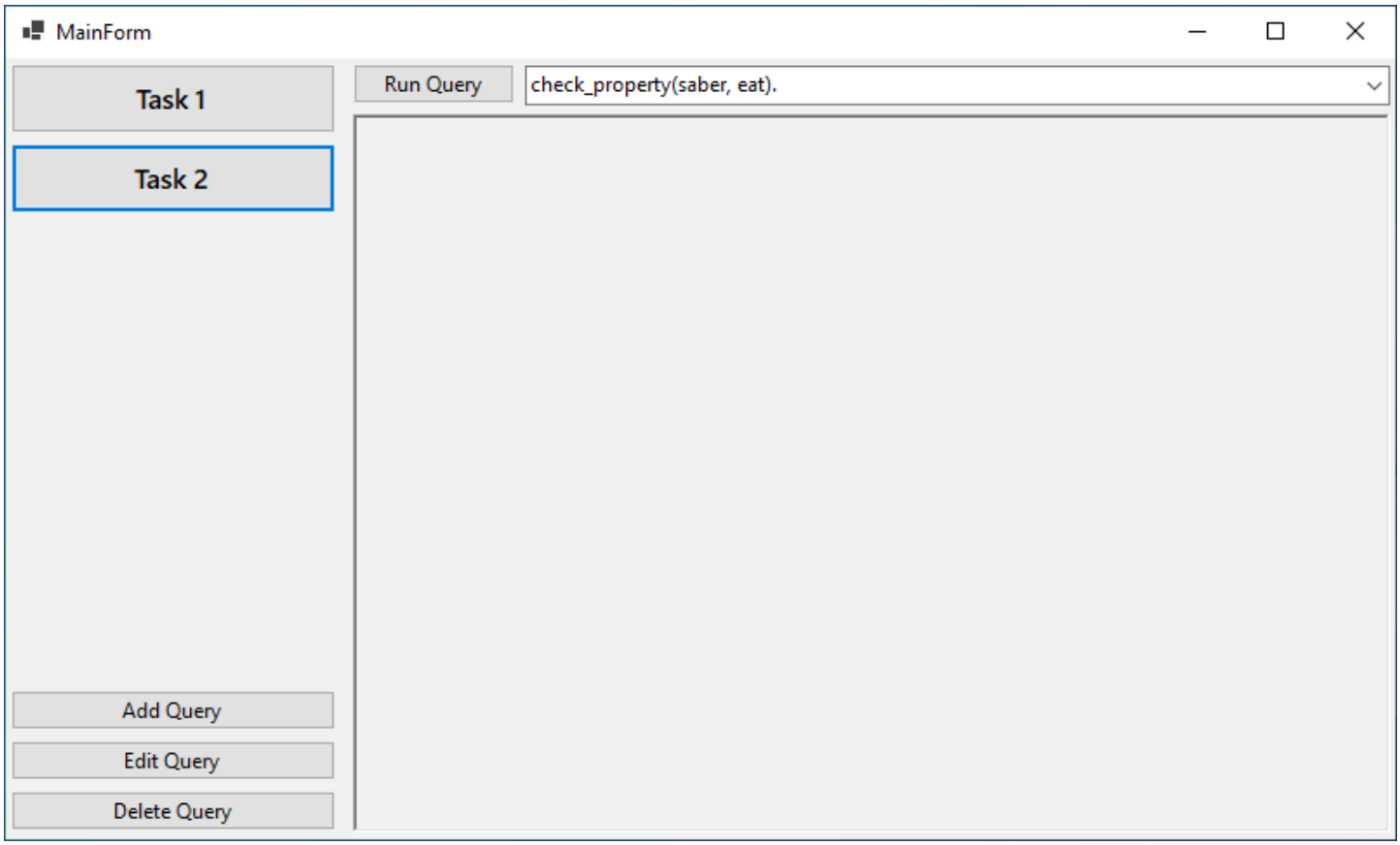


Рис. 4. Результат выбора задачи 2

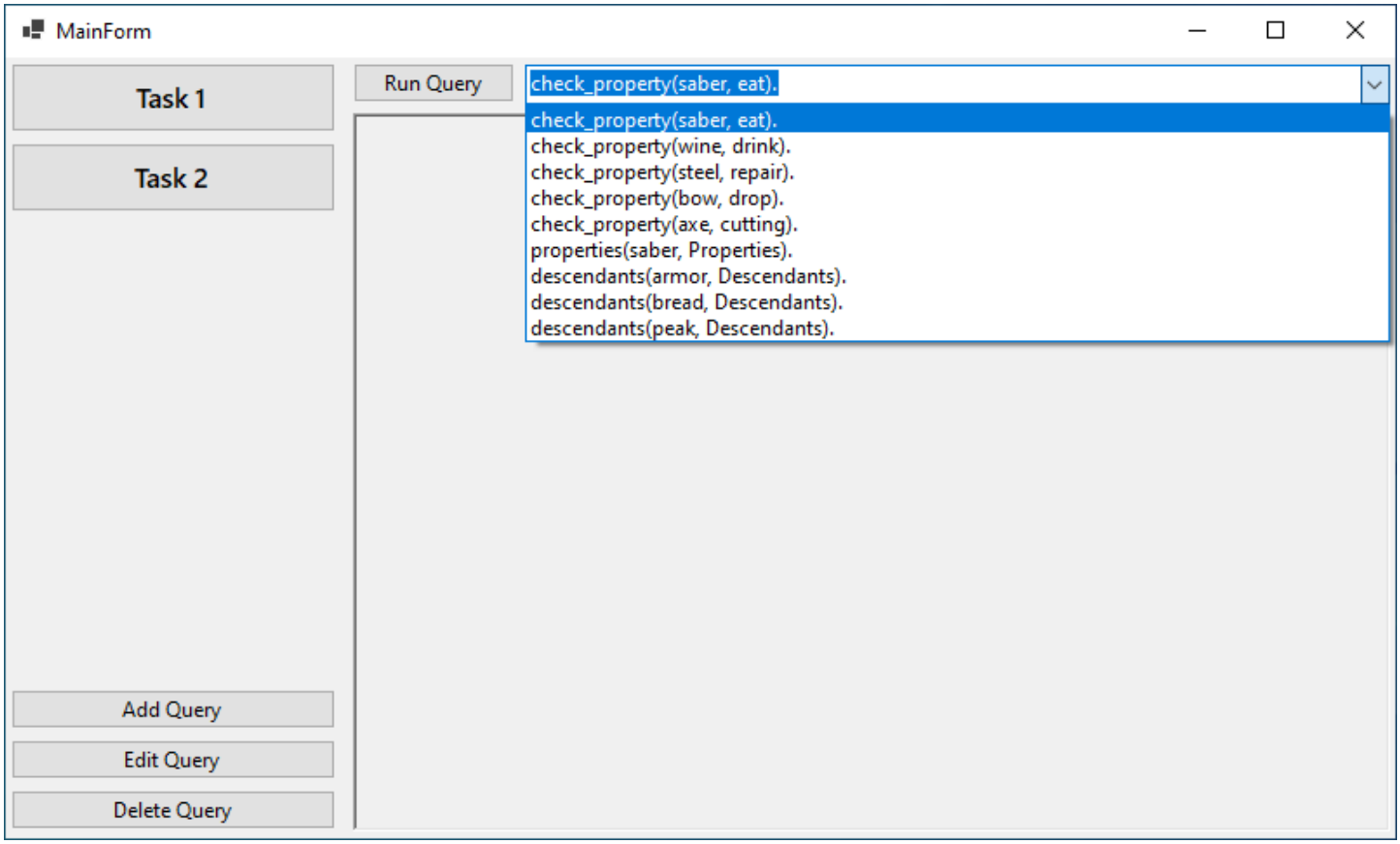


Рис. 5. Выбор запроса

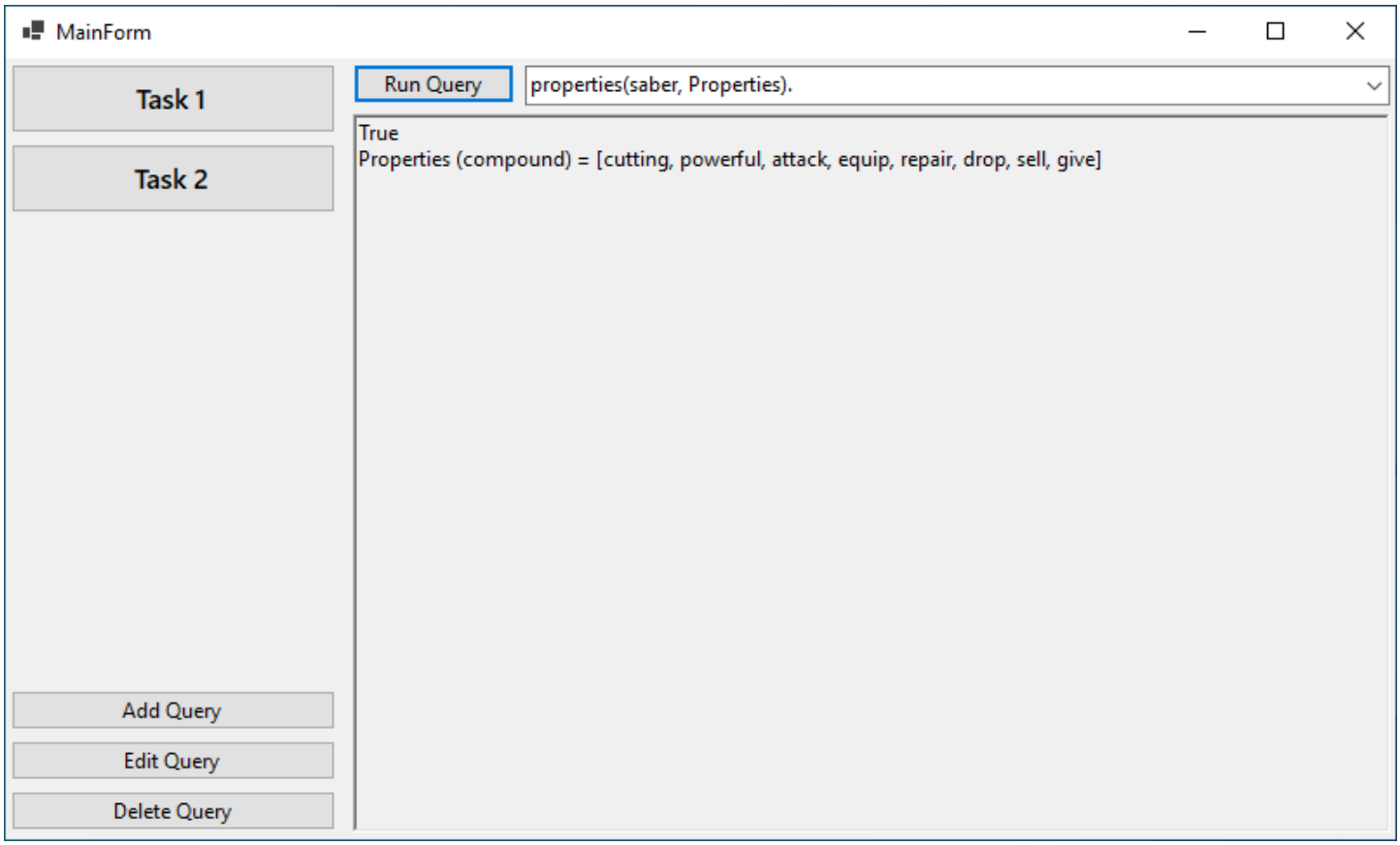
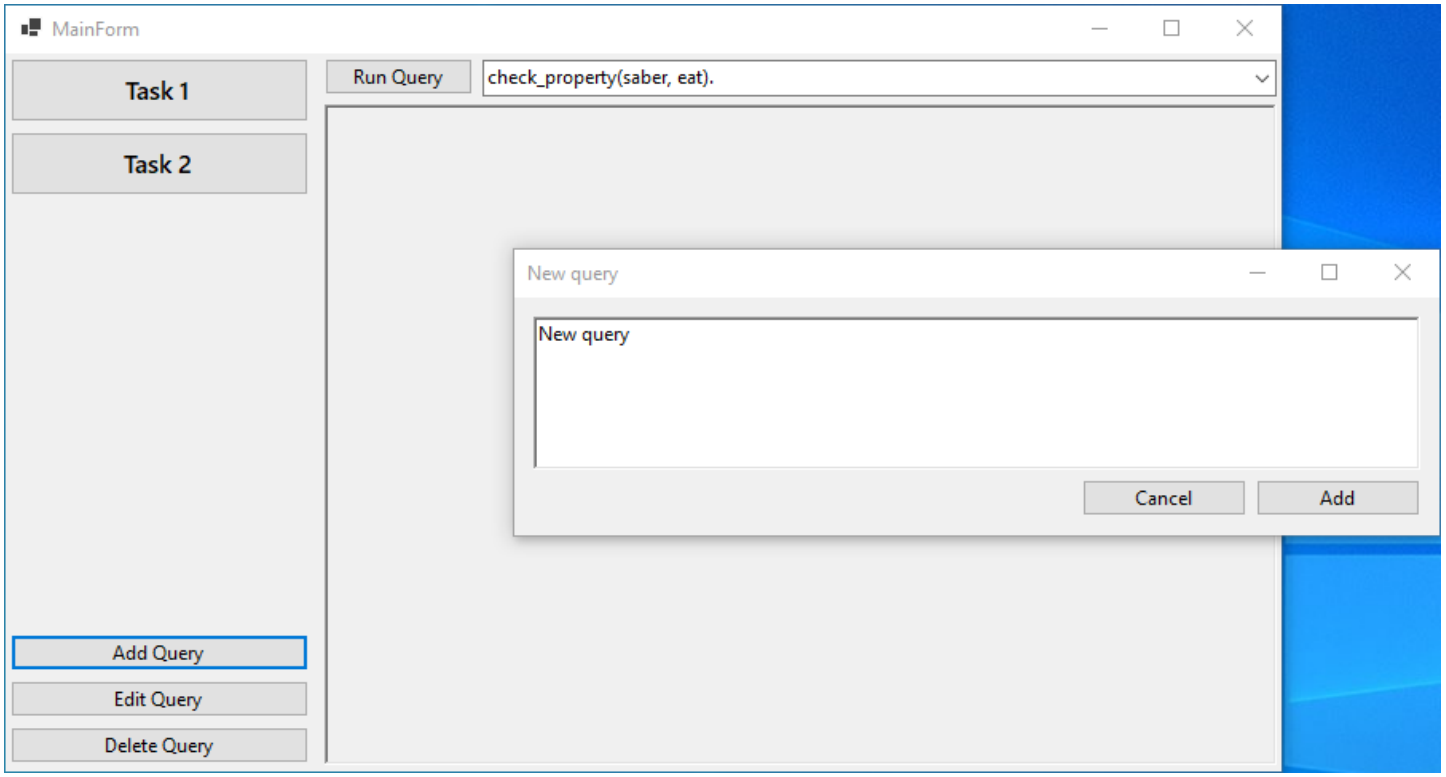


Рис. 6. Результат выполнения запроса



С полным листингом интерфейса можно ознакомиться в приложении В.

ВЫВОДЫ

В ходе выполнения лабораторной работы №9 было произведено знакомство с основами логического программирования на примере языка Prolog. Были написаны семейное древо и семантическая сеть предметов из RPG-игры. Были реализованы рекурсивные правила.

В результаты выполнения лабораторной работы №9 было разработано приложение на языке высокого уровня C#, предоставляющее пользователю удобный интерфейс взаимодействия (добавление, изменение и удаление запросов) с разработанной на языке логического программирования Prolog базой знаний.

ПРИЛОЖЕНИЕ А. ЛИСТИНГ ПРОГРАММЫ К ЗАДАНИЮ 1

```
/* Родители: */
parent("Mikhail Fedorovich", "Alexey Mikhailovich").
parent("Evdokia Lukyanova", "Alexey Mikhailovich").

parent("Alexey Mikhailovich", "Feodor Alekseevich").
parent("Alexey Mikhailovich", "Sofya Alekseevna").
parent("Alexey Mikhailovich", "John 5 Alekseevich").
parent("Alexey Mikhailovich", "Peter 1 Alekseevich").
parent("Maria Ilyinichna Miloslavskaya", "Feodor Alekseevich").
parent("Maria Ilyinichna Miloslavskaya", "Sofya Alekseevna").
parent("Maria Ilyinichna Miloslavskaya", "John 5 Alekseevich").
parent("Natalia Kirillovna Naryshkina", "Peter 1 Alekseevich").

parent("John 5 Alekseevich", "Ekaterina Ioannovna").
parent("John 5 Alekseevich", "Anna Ioannovna").
parent("Praskovya Fedorovna Saltykova", "Ekaterina Ioannovna").
parent("Praskovya Fedorovna Saltykova", "Anna Ioannovna").
parent("Peter 1 Alekseevich", "Alexey Petrovich").
parent("Peter 1 Alekseevich", "Anna Petrovna").
parent("Peter 1 Alekseevich", "Elizabeth Petrovna").
parent("Evdokia Feodorovna Lopukhina", "Alexey Petrovich").
parent("Ekaterina 1 Alekseevna", "Anna Petrovna").
parent("Ekaterina 1 Alekseevna", "Elizabeth Petrovna").

parent("Ekaterina Ioannovna", "Anna Leopoldovna").
parent("Karl Leopold of Mecklenburg-Schwerin", "Anna Leopoldovna").
parent("Alexey Petrovich", "Peter 2 Alekseevich").
parent("Natalia Alekseevna Braunschweig-Wolfenbuttelskaya", "Peter 2 Alekseevich").
parent("Anna Petrovna", "Peter 3 Fyodorovich").
parent("Karl Friedrich of Holstein-Gottorp", "Peter 3 Fyodorovich").

parent("Anna Leopoldovna", "John 6 Antonovich").
parent("Anton Ulrich Brauschweig-Wolfenbuttelsky", "John 6 Antonovich").
parent("Peter 3 Fyodorovich", "Pavel 1 Petrovich").
parent("Ekaterina 2 Alekseevna", "Pavel 1 Petrovich").

parent("Pavel 1 Petrovich", "Alexander 1 Pavlovich").
parent("Pavel 1 Petrovich", "Nikolai 1 Pavlovich").
parent("Maria Feodorovna of Württemberg", "Alexander 1 Pavlovich").
parent("Maria Feodorovna of Württemberg", "Nikolai 1 Pavlovich").

parent("Nikolai 1 Pavlovich", "Alexander 2 Nikolaevich").
parent("Alexandra Feodorovna Prusskaya", "Alexander 2 Nikolaevich").

parent("Alexander 2 Nikolaevich", "Alexander 3 Alexandrovich").
parent("Maria Alexandrovna of Hesse", "Alexander 3 Alexandrovich").

parent("Alexander 3 Alexandrovich", "Nikolai 2 Alexandrovich").
parent("Maria Feodorovna of Denmark", "Nikolai 2 Alexandrovich").

/* Женщины: */
woman("Evdokia Lukyanova").

woman("Maria Ilyinichna Miloslavskaya").
woman("Natalia Kirillovna Naryshkina").

woman("Sofya Alekseevna").
woman("Marfa Matveevna Apraksina").
woman("Praskovya Fedorovna Saltykova").
woman("Evdokia Feodorovna Lopukhina").
woman("Ekaterina 1 Alekseevna").

woman("Ekaterina Ioannovna").
woman("Anna Ioannovna").
woman("Natalia Alekseevna Braunschweig-Wolfenbuttelskaya").
woman("Anna Petrovna").
woman("Elizabeth Petrovna").

woman("Anna Leopoldovna").
woman("Ekaterina 2 Alekseevna").

woman("Natalia Alekseevna Hesse-Darmstadt").
```

```
woman("Maria Feodorovna of Württemberg").

woman("Elizabeth Alekseevna").
woman("Alexandra Feodorovna Prusskaya").

woman("Maria Alexandrovna of Hesse").

woman("Maria Feodorovna of Denmark").

woman("Alexandra Feodorovna of Hesse").

/* Мужчины: */
man("Mikhail Fedorovich").

man("Alexey Mikhailovich").

man("Feodor Alekseevich").
man("John 5 Mikhailovich").
man("Peter 1 Alekseevich").

man("Karl Leopold of Mecklenburg-Schwerin").
man("Frederick William of Courland").
man("Alexey Petrovich").
man("Karl Friedrich of Holstein-Gottorp").

man("Anton Ulrich Brauschweig-Wolfenbuttelsky").
man("Peter 2 Alekseevich").
man("Peter 3 Fyodorovich").

man("John 6 Antonovich").
man("Pavel 1 Petrovich").

man("Alexander 1 Pavlovich").
man("Nikolai 1 Pavlovich").

man("Alexander 2 Nikolaevich").

man("Alexander 3 Alexandrovich").

man("Nikolai 2 Alexandrovich").

/* Брак: */
married("Mikhail Fedorovich", "Evdokia Lukyanova").

married("Alexey Mikhailovich", "Maria Ilyinichna Miloslavskaya").
married("Alexey Mikhailovich", "Natalia Kirillovna Naryshkina").

married("Feodor Alekseevich", "Marfa Matveevna Apraksina").
married("John 5 Alekseevich", "Praskovya Fedorovna Saltykova").
married("Peter 1 Alekseevich", "Evdokia Feodorovna Lopukhina").
married("Peter 1 Alekseevich", "Ekaterina 1 Alekseevna").
married("Karl Leopold of Mecklenburg-Schwerin", "Ekaterina Ioannovna").
married("Frederick William of Courland", "Anna Ioannovna").
married("Alexey Petrovich", "Natalia Alekseevna Braunschweig-Wolfenbuttelskaya").
married("Karl Friedrich of Holstein-Gottorp", "Anna Petrovna").

married("Anton Ulrich Brauschweig-Wolfenbuttelsky", "Anna Leopoldovna").
married("Peter 3 Fyodorovich", "Ekaterina 2 Alekseevna").

married("Pavel 1 Petrovich", "Natalia Alekseevna Hesse-Darmstadt").
married("Pavel 1 Petrovich", "Maria Feodorovna of Württemberg").
married("Alexander 1 Pavlovich", "Elizabeth Alekseevna").
married("Nikolai 1 Pavlovich", "Alexandra Feodorovna Prusskaya").

married("Alexander 2 Nikolaevich", "Maria Alexandrovna of Hesse").

married("Alexander 3 Alexandrovich", "Maria Feodorovna of Denmark").

married("Nikolai 2 Alexandrovich", "Alexandra Feodorovna of Hesse").

/* Правила: */
```

```
check_married(X, Y) :-              /* Женаты ли? */
       married(X, Y);
       married(Y, X).
father(X, Y) :-                     /* Отец */
       parent(X, Y),
       man(X).
mother(X, Y) :-                     /* Мать */
       parent(X, Y),
       woman(X).
brother(X, Y) :-                    /* Брат */
       parent(Z, X),
       parent(Z, Y),
       man(X),
       X \= Y.
sister(X, Y) :-                     /* Сестра */
       parent(Z, X),
       parent(Z, Y),
       woman(X),
       X \= Y.
son(X, Y) :-                        /* Сын */
       parent(Y, X),
       man(X).
daughter(X, Y) :-                   /* Дочь */
       parent(Y, X),
       woman(X).
husband(X, Y) :-                    /* Муж */
       check_married(X, Y),
       man(X).
wife(X, Y) :-                       /* Жена */
       check_married(X, Y),
       woman(X).
grandfather(X, Y) :-                /* Дедушка */
       parent(X, Z),
       parent(Z, Y),
       man(X).
grandmother(X, Y) :-                /* Бабушка */
       parent(X, Z),
       parent(Z, Y),
       woman(X).

grandson(X, Y) :-                   /* Внук */
       parent(Z, Y),
       parent(X, Z),
       man(Y).
granddaughter(X, Y) :-              /* Внучка */
       parent(Z, Y),
       parent(X, Z),
       woman(Y).
predecessor(X, Y) :-                /* Ближний предок */
       parent(X, Y).
predecessor(X, Y) :-                /* Дальний (рекурсивный) предок */
       parent(X, Z),
       predecessor(Z, Y).

/* Примеры использования:
       ?- father("Karl Leopold of Mecklenburg-Schwerin", "Anna Leopoldovna").                    true
       ?- mother("Natalia Alekseevna Braunschweig-Wolfenbuttelskaya", "Peter 2 Alekseevich").    true
       ?- mother("Natalia Alekseevna Braunschweig-Wolfenbuttelskaya", "Nikolai 2 Alexandrovich"). false
       ?- brother("Feodor Alekseevich", "John 5 Alekseevich").                                   true
       ?- sister("Ekaterina Ioannovna", "Anna Ioannovna").                                       true
       ?- son("Nikolai 2 Alexandrovich", "Alexander 3 Alexandrovich").                           true
       ?- daughter("Elizabeth Petrovna", "Alexander 3 Alexandrovich").                           false
       ?- daughter("Elizabeth Petrovna", "Ekaterina 1 Alekseevna").                              true
       ?- husband("Alexander 1 Pavlovich", "Elizabeth Alekseevna").                              true
       ?- wife("Feodor Alekseevich", "Marfa Matveevna Apraksina").                               true
       ?- grandson("Alexander 2 Nikolaevich", "Nikolai 2 Alexandrovich").                        true
       ?- granddaughter("Alexey Mikhailovich", "Anna Ioannovna").                                true
       ?- predecessor("Peter 1 Alekseevich", "Alexey Petrovich").                                true
*/
```

ПРИЛОЖЕНИЕ Б. ЛИСТИНГ ПРОГРАММЫ К ЗАДАНИЮ 2

```
/* Что от чего наследуется: */

connect(edible, object).                /* (съедобный, объект) */
connect(inedible, object).              /* (несъедобный, объект) */

connect(food, edible).                  /* (еда, съедобный) */
connect(beverage, edible).              /* (напиток, съедобный) */
connect(equipment, inedible).           /* (снаряжение, несъедобный) */

connect(raw, food).                     /* (сырая, еда) */
connect(meal, food).                    /* (блюдо, еда) */
connect(non_alcoholic, beverage).       /* (безалкогольное, напиток) */
connect(alcohol, beverage).             /* (алкоголь, напиток) */
connect(potion, beverage).              /* (зелье, напиток) */
connect(armor, equipment).              /* (броня, снаряжение) */
connect(weapon, equipment).             /* (оружие, снаряжение) */

connect(berry, raw).                    /* (ягода , сырая) */
connect(meat, raw).                     /* (мясо , сырая) */
connect(mushroom, raw).                 /* (гриб , сырая) */
connect(steak, meal).                   /* (стейк , блюдо) */
connect(pie, meal).                     /* (пирог , блюдо) */
connect(soup, meal).                    /* (суп , блюдо) */
connect(bread, meal).                   /* (хлеб , блюдо) */
connect(water, non_alcoholic).          /* (вода, безалкогольное) */
connect(milk, non_alcoholic).           /* (молоко, безалкогольное) */
connect(juice, non_alcoholic).          /* (сок, безалкогольное) */
connect(beer, alcohol).                 /* (пиво, алкоголь) */
connect(wine, alcohol).                 /* (вино, алкоголь) */
connect(brandy, alcohol).               /* (бренди, алкоголь) */
connect(whiskey, alcohol).              /* (виски, алкоголь) */
connect(vodka, alcohol).                /* (водка, алкоголь) */
connect(strength, potion).              /* (сила, зелье) */
connect(speed, potion).                 /* (скорость, зелье) */
connect(agility, potion).               /* (ловкость, зелье) */
connect(stamina, potion).               /* (выносливость, зелье) */
connect(health, potion).                /* (здоровье, зелье) */
connect(manna, potion).                 /* (манна, зелье) */
connect(heavy, armor).                  /* (тяжелая, броня) */
connect(light, armor).                  /* (легкая, броня) */
connect(melee, weapon).                 /* (ближнего боя, оружие) */
connect(ranged, weapon).                /* (дальнего боя, оружие) */

connect(steel, heavy).                  /* (стальная, тяжелая) */
connect(hauberk, light).                /* (кольчуга, легкая) */
connect(leathern, light).               /* (кожаная, легкая) */
connect(short, melee).                  /* (короткое, ближнего боя) */
connect(long, melee).                   /* (длинное, ближнего боя) */
connect(bow, ranged).                   /* (лук, дальнего боя) */
connect(crossbow, ranged).              /* (арбалет, дальнего боя) */

connect(dagger, short).                 /* (кинжал, короткое) */
connect(knife, short).                  /* (нож, короткое) */
connect(one_handed, long).              /* (одноручное , длинное) */
connect(two_handed, long).              /* (двуручное , длинное) */

connect(broadsword, one_handed).        /* (тесак, одноручное) */
connect(hatchet, one_handed).           /* (топорик, одноручное) */
connect(baton, one_handed).             /* (дубинка, одноручное) */
connect(saber, one_handed).             /* (сабля, одноручное) */
connect(sword, one_handed).             /* (меч, одноручное) */
connect(peak, two_handed).              /* (пика, двуручное) */
connect(battle_axe, two_handed).        /* (секира, двуручное) */
connect(scythe, two_handed).            /* (коса, двуручное) */
connect(halberd, two_handed).           /* (алебарда, двуручное) */

/* Свойства: */
property(object, drop).                 /* (предмет, выбросить) */
property(object, sell).                 /* (предмет, продать) */
property(object, give).                 /* (предмет, отдать) */

property(edible, eat).                  /* (съедобный, есть) */
```

```
property(beverage, drink).                /* (напиток, выпить) */
property(equipment, equip).               /* (снаряжение, экипировать) */
property(equipment, repair).              /* (снаряжение, чинить) */

property(raw, cook ).                     /* (сырая, приготовить) */
property(armor, protect).                 /* (броня, защищать) */
property(weapon, attack).                 /* (оружие, атаковать) */
property(heavy, reliable).                /* (тяжелая, надежная) */
property(light, comfortable).             /* (легкая, комфортная) */

property(ranged, shoot).                  /* (дальнего боя, стрелять) */

property(short, comfortable).             /* (короткое, удобный) */
property(long, powerful).                 /* (длинное, мощный) */
property(crossbow, reload).               /* (арбалет, перезарядить) */

property(dagger, cutting).                /* (кинжал, резать) */
property(dagger, stabbing).               /* (кинжал, колоть) */
property(knife, cutting).                 /* (нож, резать) */
property(knife, stabbing).                /* (нож, колоть) */
property(broadsword, cutting).            /* (тесак, резать) */
property(broadsword, chopping).           /* (тесак, рубить) */
property(hatchet, chopping).              /* (топорик, рубить) */
property(baton, hitting).                 /* (дубинка, бить) */
property(saber, cutting).                 /* (сабля, резать) */
property(sword, cutting).                 /* (меч, резать) */
property(sword, stabbing).                /* (меч, колоть) */
property(sword, chopping).                /* (меч, рубить) */
property(peak, stabbing).                 /* (пика, колоть) */
property(battle_axe, chopping).           /* (секира, рубить) */
property(scythe, cutting).                /* (коса, резать) */
property(halberd, chopping).              /* (алебарда, рубить) */

/* Наследование */
inherit(X, Z) :-                          /* X наследует свойство Z, если: */
       connect(X, Y),                     /* X соединен с Y */
       property_too(Y, Z).                /* и этот Y обладает темже свойством Z*/

property_too(X, Z) :-                     /* X также обладает свойством Z, если: */
       property(X, Z);                    /* X обладает свойством Z */
       inherit(X, Z).                     /* или X наследует свойство Z */

descendant(X, Y) :-                       /* X потомок для Y, если: */
       connect(X, Y).                     /* X связан с Y */
descendant(X, Y) :-                       /* X потомок для Y, если: */
       connect(X, Z),                     /* X связан с Z */
       descendant(Z, Y).                  /* и этот Z потомок для Y */

check_property(X, Z) :-
       property_too(X, Z).

properties(X, Properties) :-
       findall(Property, check_property(X, Property), Properties).

descendants(Y, Descendants) :-
       findall(Descendant, descendant(Descendant, Y), Descendants).

/* Примеры использования:
       ?- check_property(saber, eat).             false
       ?- check_property(wine, drink).            true
       ?- check_property(steel, repair).          true
       ?- check_property(bow, drop).              true
       ?- check_property(axe, cutting).           false

       ?- properties(saber, Properties).          Properties = [cutting, powerful, attack, equip, repair,
drop, sell, give].

       ?- descendants(armor, Descendants).        Descendants = [heavy, light, steel, hauberk, leathern].
       ?- descendants(bread, Descendants).        Descendants = [].
       ?- descendants(peak, Descendants).         Descendants = [].
*/
```

ПРИЛОЖЕНИЕ В. ЛИСТИНГ ПРОГРАММЫ К ЗАДАНИЮ 3

Form1.cs

```
using Prolog;

namespace PUI
{
    public partial class Form1 : Form
    {
        PrologEngine prolog = new(persistentCommandHistory: false);
        string task1 = "E:\\Projects VisualStudio\\TRPO\\PUI\\PUI\\Task1.pl";
        string task2 = "E:\\Projects VisualStudio\\TRPO\\PUI\\PUI\\Task2.pl";

        string[] queries1 = new string[]
        {
                        "father(\"Karl Leopold of Mecklenburg-Schwerin\", \"Anna
Leopoldovna\").",
                        "mother(\"Natalia Alekseevna Braunschweig-Wolfenbuttelskaya\", \"Peter 2
Alekseevich\").",
                        "mother(\"Natalia Alekseevna Braunschweig-Wolfenbuttelskaya\", \"Nikolai
2 Alexandrovich\").",
                        "brother(\"Feodor Alekseevich\", \"John 5 Alekseevich\").",
                        "sister(\"Ekaterina Ioannovna\", \"Anna Ioannovna\").\t",
                        "son(\"Nikolai 2 Alexandrovich\", \"Alexander 3 Alexandrovich\").",
                        "daughter(\"Elizabeth Petrovna\", \"Alexander 3 Alexandrovich\").",
                        "daughter(\"Elizabeth Petrovna\", \"Ekaterina 1 Alekseevna\").\t",
                        "husband(\"Alexander 1 Pavlovich\", \"Elizabeth Alekseevna\").",
                        "wife(\"Feodor Alekseevich\", \"Marfa Matveevna Apraksina\").",
                        "grandson(\"Alexander 2 Nikolaevich\", \"Nikolai 2 Alexandrovich\").",
                        "granddaughter(\"Alexey Mikhailovich\", \"Anna Ioannovna\").",
                        "predecessor(\"Peter 1 Alekseevich\", \"Alexey Petrovich\").",
        };
        string[] queries2 = new string[]
        {
                        "check_property(saber, eat).",
                        "check_property(wine, drink).",
                        "check_property(steel, repair).",
                        "check_property(bow, drop).",
                        "check_property(axe, cutting).",
                        "properties(saber, Properties).",
                        "descendants(armor, Descendants).",
                        "descendants(bread, Descendants).",
                        "descendants(peak, Descendants)."
        };
        bool isTask1 = false;

        SolutionSet solutions;
        public Form1()
        {
            InitializeComponent();
            try
            {
                prolog.Consult(task1);
                prolog.Consult(task2);
            }
            catch (Exception ex)
            {
                MessageBox.Show($"Программа прервана: {ex.Message}", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }
            resultTextBox.ReadOnly = true;
            addButton.Enabled = false;
            editButton.Enabled = false;
            deleteButton.Enabled = false;
            runButton.Enabled = false;
            queryComboBox.Enabled = false;
        }
```

```
        private void task1Button_Click(object sender, EventArgs e)
        {
            queryComboBox.Enabled = true;
            queryComboBox.Items.Clear();
            queryComboBox.Items.AddRange(queries1);
            queryComboBox.SelectedIndex = 0;
            solutions = prolog.GetAllSolutions(sourceFileName: task1, query:
queries1[queryComboBox.SelectedIndex]);
            addButton.Enabled = true;
            editButton.Enabled = true;
            deleteButton.Enabled = true;
            runButton.Enabled = true;
            isTask1 = true;
        }

        private void runButton_Click(object sender, EventArgs e)
        {
            resultTextBox.Clear();
            resultTextBox.Text += solutions.Success.ToString();
            if (solutions.Success)
            {
                resultTextBox.Text += "\n";
                for (int i = 0; i < solutions.Count; i++)
                {
                    resultTextBox.Text += solutions[i].ToString() + "\n";
                }
            }
        }

        private void queryComboBox_SelectedIndexChanged(object sender, EventArgs e)
        {
            if (isTask1)
            {
                solutions = prolog.GetAllSolutions(sourceFileName: task1, query:
queries1[queryComboBox.SelectedIndex]);
            }
            else
            {
                solutions = prolog.GetAllSolutions(sourceFileName: task2, query:
queries2[queryComboBox.SelectedIndex]);
            }

            resultTextBox.Clear();
        }
        private void task2Button_Click(object sender, EventArgs e)
        {
            queryComboBox.Enabled = true;
            queryComboBox.Items.Clear();
            queryComboBox.Items.AddRange(queries2);
            queryComboBox.SelectedIndex = 0;
            solutions = prolog.GetAllSolutions(sourceFileName: task2, query:
queries2[queryComboBox.SelectedIndex]);
            addButton.Enabled = true;
            editButton.Enabled = true;
            deleteButton.Enabled = true;
            runButton.Enabled = true;
            isTask1 = false;
        }
        private void addButton_Click(object sender, EventArgs e)
        {
            Form2 form2 = new Form2();
            form2.Text = "New query";
            form2.ShowDialog();
            if (form2.DialogResult == DialogResult.OK)
            {
                if (isTask1)
                {
                    queries1 = queries1.Append(form2.newQuery).ToArray();
                }
```

```
                else
                {
                    queries2 = queries2.Append(form2.newQuery).ToArray();
                }
                queryComboBox.Items.Clear();
                if (isTask1)
                {
                    queryComboBox.Items.AddRange(queries1);
                }
                else
                {
                    queryComboBox.Items.AddRange(queries2);
                }
                queryComboBox.SelectedIndex = 0;
                queryComboBox_SelectedIndexChanged(sender, e);
            }
        }
        private void editButton_Click(object sender, EventArgs e)
        {
            Form3 form3 = new Form3();
            form3.Text = "Edit query";
            if (isTask1)
            {
                form3.editedQuery = queries1[queryComboBox.SelectedIndex];
            }
            else
            {
                form3.editedQuery = queries2[queryComboBox.SelectedIndex];
            }
            form3.ShowDialog();
            if (form3.DialogResult == DialogResult.OK)
            {
                if (isTask1)
                {
                    queries1[queryComboBox.SelectedIndex] = form3.editedQuery;
                    queryComboBox.Items.Clear();
                    queryComboBox.Items.AddRange(queries1);
                }
                else
                {
                    queries2[queryComboBox.SelectedIndex] = form3.editedQuery;
                    queryComboBox.Items.Clear();
                    queryComboBox.Items.AddRange(queries2);
                }
                queryComboBox.SelectedIndex = 0;
                queryComboBox_SelectedIndexChanged(sender, e);
            }
        }
        private void deleteButton_Click(object sender, EventArgs e)
        {
            if (isTask1)
            {
                queries1 = queries1.Where((source, index) => index !=
queryComboBox.SelectedIndex).ToArray();
                queryComboBox.Items.Clear();
                queryComboBox.Items.AddRange(queries1);
            }
            else
            {
                queries2 = queries2.Where((source, index) => index !=
queryComboBox.SelectedIndex).ToArray();
                queryComboBox.Items.Clear();
                queryComboBox.Items.AddRange(queries2);
            }
            queryComboBox.SelectedIndex = 0;
            queryComboBox_SelectedIndexChanged(sender, e);
        }
    }
}
```

Form2.cs

```
namespace PUI
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
            newQueryTextBox.Text = "Your query";
        }

        public string newQuery { get; set; } = null;

        private void addButton_Click(object sender, EventArgs e)
        {
            if (newQueryTextBox.Text == "Your query" || newQueryTextBox.Text == "")
            {
                MessageBox.Show("Введите запрос", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            newQuery = newQueryTextBox.Text;
        }

        private void cancelButton_Click(object sender, EventArgs e)
        {
            return;
        }

        private void Form2_Load(object sender, EventArgs e)
        {

        }
    }
}
```

Form3.cs

```
namespace PUI
{
    public partial class Form3 : Form
    {
        public string editedQuery { get; set; } = null;
        public Form3()
        {
            InitializeComponent();
            editQueryTextBox.Text = editedQuery;
        }

        private void addButton_Click(object sender, EventArgs e)
        {
            if (editQueryTextBox.Text == editedQuery || editQueryTextBox.Text == "")
            {
                MessageBox.Show("Введите запрос", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            editedQuery = editQueryTextBox.Text;
        }

        private void cancelButton_Click(object sender, EventArgs e)
        {
            return;
        }
    }
}
```