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

# Лабораторная работа №4

по теме “Разработка приложения SWI-пролог”

по курсу

«Теория разработки программного обеспечения»

Выполнил: студент группы
А-07м-23
Балашов С.А.
Проверила: Раскатова М.В.

Москва, 2024 г.

**Содержание**

## 1. Задание

1.1. 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.

1.2. 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.

1.3. На ЯП разработать инструментальную часть + графический интерфейс.

## 2. Разработка

1. В качестве семейного древа было выбрано семейное древо из Ветхого заветка, начиная от Адама и заканчивая сыновьями Ноя. Планируется реализация фактов для образования семейного древа и правил для проверки вытекающих фактов.

2. В качестве предметной области для семантической сети был выбран класс живых существ. Планируется реализация фактов для установки связей наследования между классами и присвоения им свойств и правил для проверки вытекающих фактов.

3. 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.

## 3. Пример разработанного дерева семейных отношений

На рис.1 представлено дерево семейных отношений.

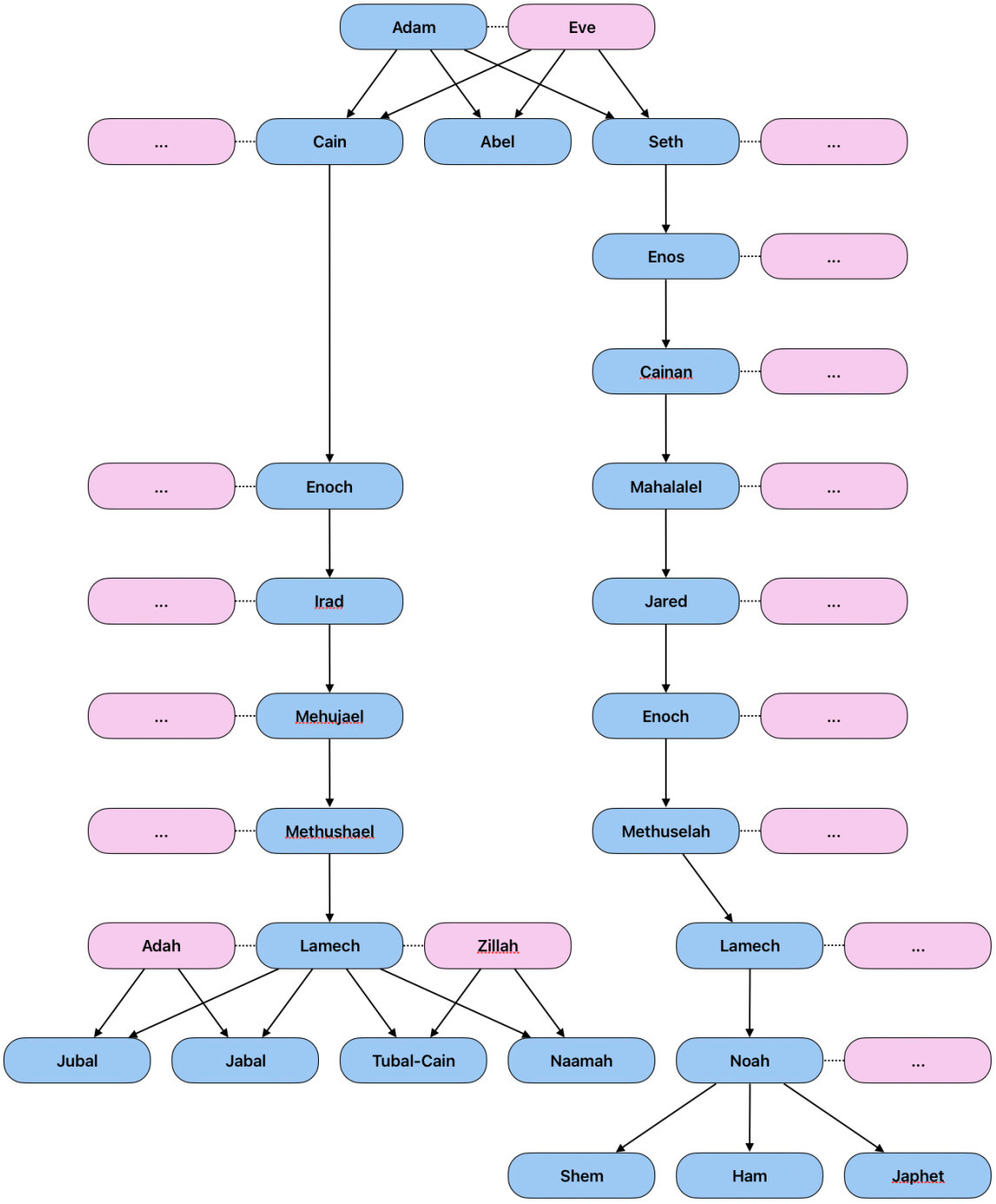


Рис. 1. Дерево семейных отношений

## 4. Пример семантической сети

На рис.2 представлена семантическая сеть живых существ

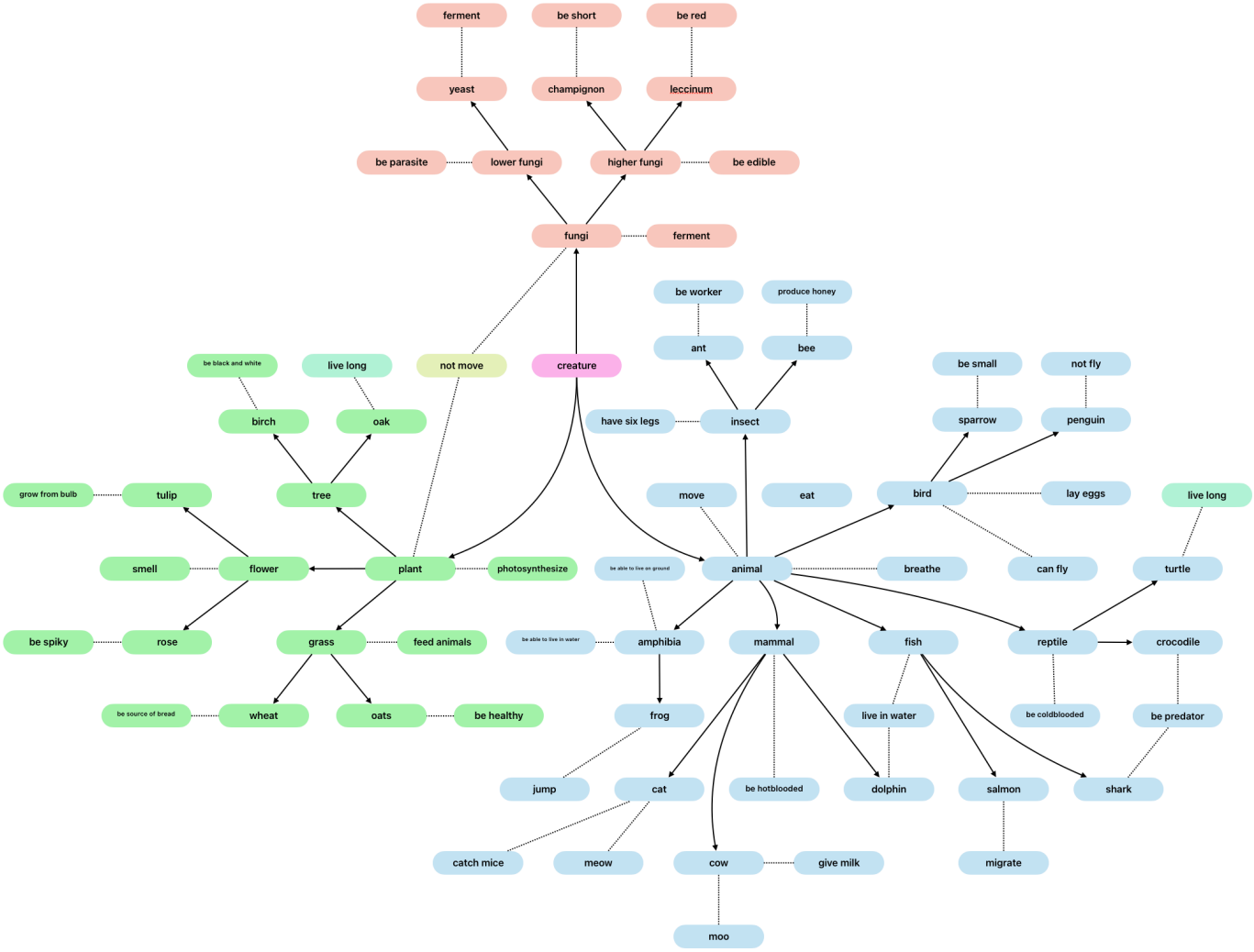


Рис. 2. Семантическая сеть живых существ

## 5. Разработанные факты и правила

1. Для семейного древа были разработаны следующие факты и правила:

1.1. parent(X, Y). - Факт родительства X по отношению к Y.

1.2. woman(X). - Факт принадлежности X к женскому полу.

1.3. man(X). - Факт принадлежности X к мужскому полу.

1.4. married(X, Y). - Факт брака между X и Y.

1.5. check_married(X, Y) :- married(X, Y), married(Y, X). - Правило проверки факта брака между X и Y.

1.6. father(X, Y) :- parent(X, Y), man(X). - Правило проверки, является ли X отцом для Y.

1.7. mother(X, Y) :- parent(X, Y), woman(X). - Правило проверки, является ли X матерью для Y.

1.8. brother(X, Y) :- parent(Z, X), parent(Z, Y), man(X), X \= Y. - Правило проверки, является ли X братом для Y.

1.9. sister(X, Y) :- parent(Z, X), parent(Z, Y), woman(X), X \= Y. - Правило проверки, является ли X сестрой для Y.

1.10. son(X, Y) :- parent(Y, X), man(X). - Правило проверки, является ли X братом для Y.

1.11. daughter(X, Y) :- parent(Y, X), woman(X). - Правило проверки, является ли X сестрой для Y.

1.12. husband(X, Y) :- check_married(X, Y), man(X). - Правило проверки, является ли X мужем для Y.

1.13. wife(X, Y) :- check_married(X, Y), woman(X). - Правило проверки, является ли X женой для Y.

1.14. uncle(X, Y) :- parent(Z, Y), brother(X, Z). - Правило проверки, является ли X дядей для Y.

1.15. aunt(X, Y) :- parent(Z, Y), sister(X, Z). - Правило проверки, является ли X тётей для Y.

1.16. grandfather(X, Y) :- parent(X, Z), parent(Z, X), man(X). - Правило проверки, является ли X дедом для Y.

1.17. grandmother(X, Y) :- parent(X, Z), parent(Z, X), woman(X). - Правило проверки, является ли X бабушкой для Y.

1.18. grandson(X, Y) :- parent(Z, Y), parent(X, Z), man(Y). - Правило проверки, является ли Y внуком для X.

1.19. granddaughter(X, Y) :- parent(Z, Y), parent(X, Z), woman(Y). - Правило проверки, является ли Y внучкой для X.

1.20. predecessor(X, Y) :- parent(X, Y). predecessor(X, Y) :- parent(X, Z), predecessor(Z, Y). - Рекурсивное правило проверки, является ли X предком для Y.

2. Для семантической сети были разработаны следующие факты и правила:

2.1. being(X, Y) - Факт принадлежности X к классу Y.

2.2. does(X, Y) - Факт свойства Y у X.

2.3. inherit(X, Z) :- being(X, Y), does_too(Y, Z). - Правило наследования принадлежности классу.

2.4. does_too(X, Z) :- does(X, Z); inherit(X, Z). - Правило наследования свойств.

2.5. ancestor(X, Y) :- being(X, Y). ancestor(X, Y) :- being(X, Z), ancestor(Z, Y). - Рекурсивное правило проверки, является ли X наследным от Y.

2.6. check_property(X, Z) :- does_too(X, Z). - Правило проверки свойства Z у X.

2.7. properties(X, Properties) :- findall(Property, check_property(X, Property), Properties). - Правило поиска всех свойств X.

2.8. ancestors(Y, Ancestors) :- findall(Ancestor, ancestor(Ancestor, Y), Ancestors). - Правило поиска все наследных классов для Y.

## 6. Результаты

В рамках задания 3 было разработано настольное приложение с графическим интерфейсом на языке C# для удобного просмотра заданий 1 и 2. При помощи этой программы можно выбрать задание, запустить один из запросов на выбор, добавить свой запрос или изменить и удалить существующий запрос. На рис. 3-14 представлено взаимодействие с программой. Листинг программ представлен в приложениях А, Б, В.

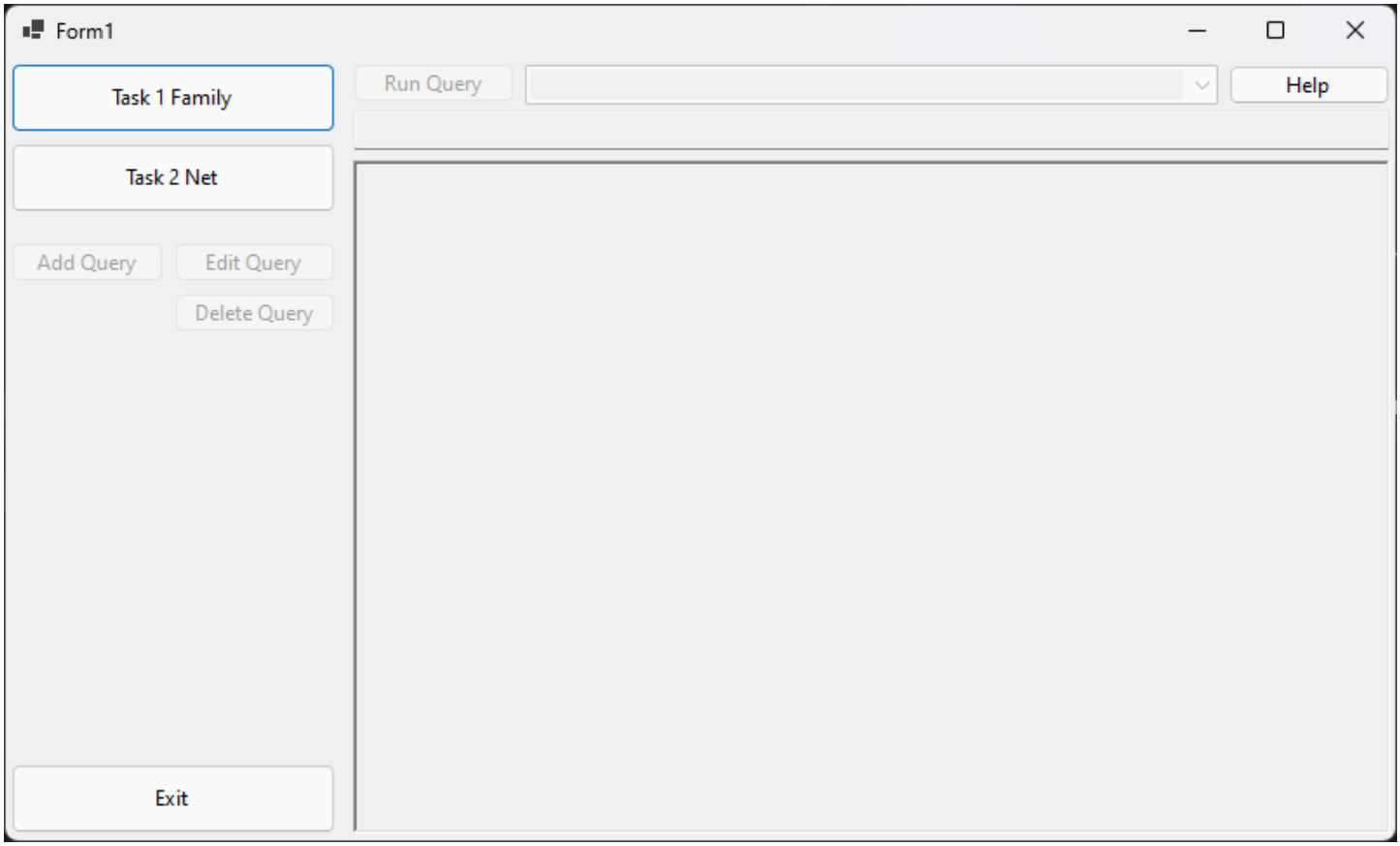


Рис. 3. Интерфейс приложения сразу после запуска.

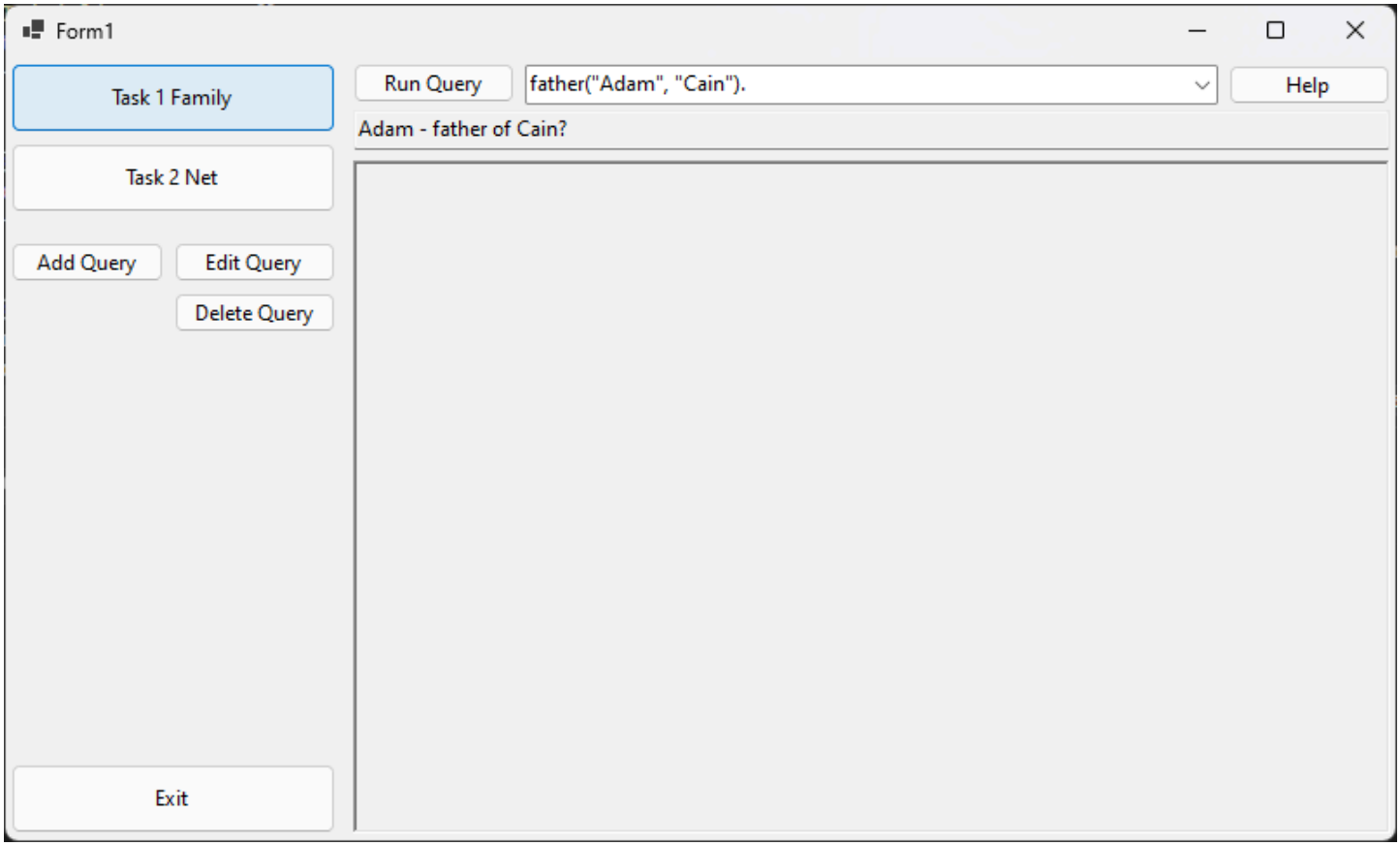


Рис. 4. Интерфейс приложения после выбора задания.

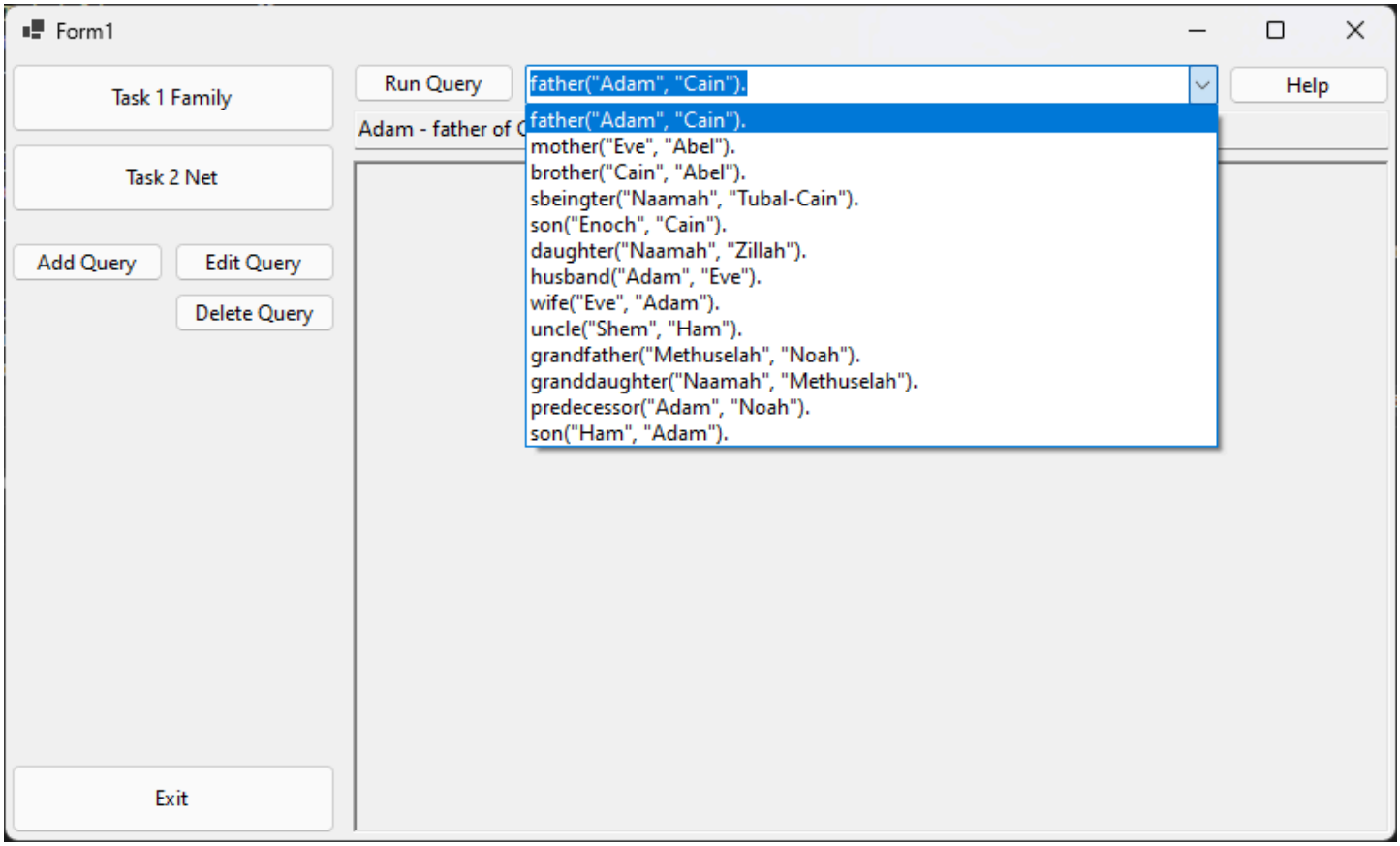


Рис. 5. Интерфейс приложения при выборе запроса.

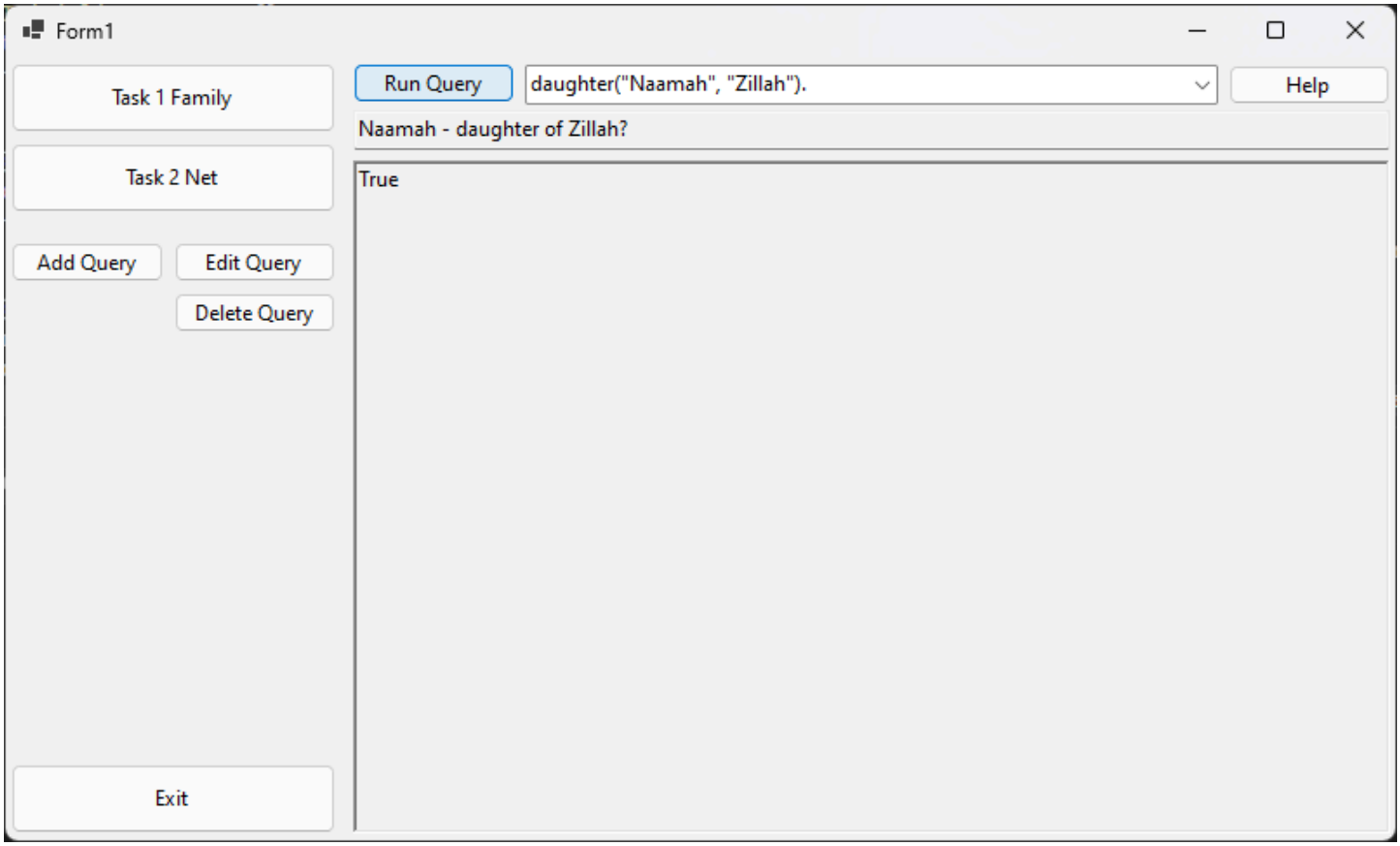


Рис. 6. Интерфейс приложения при запуске выбранного запроса.

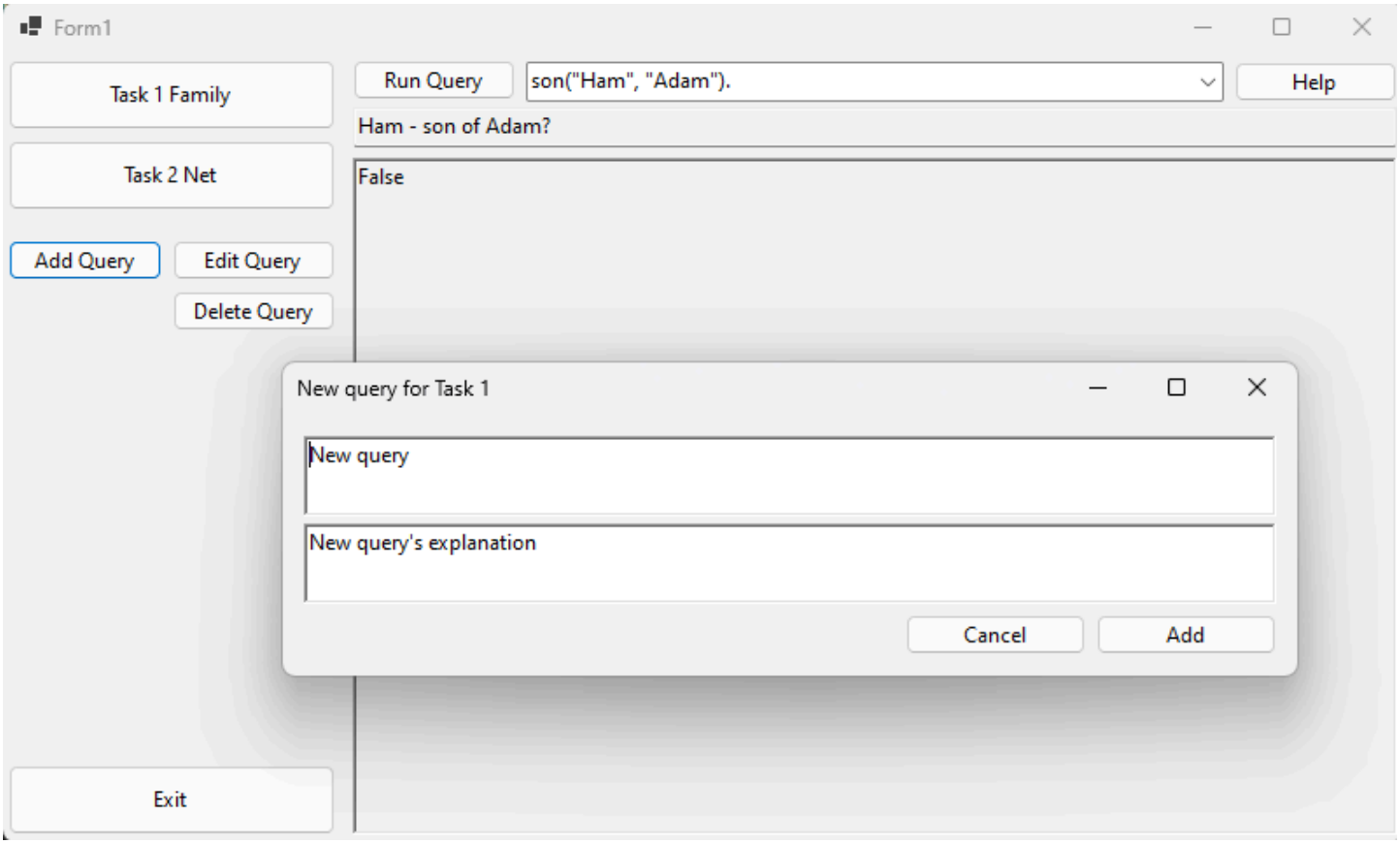


Рис. 7. Интерфейс приложения при нажатии кнопки добавления нового запроса.

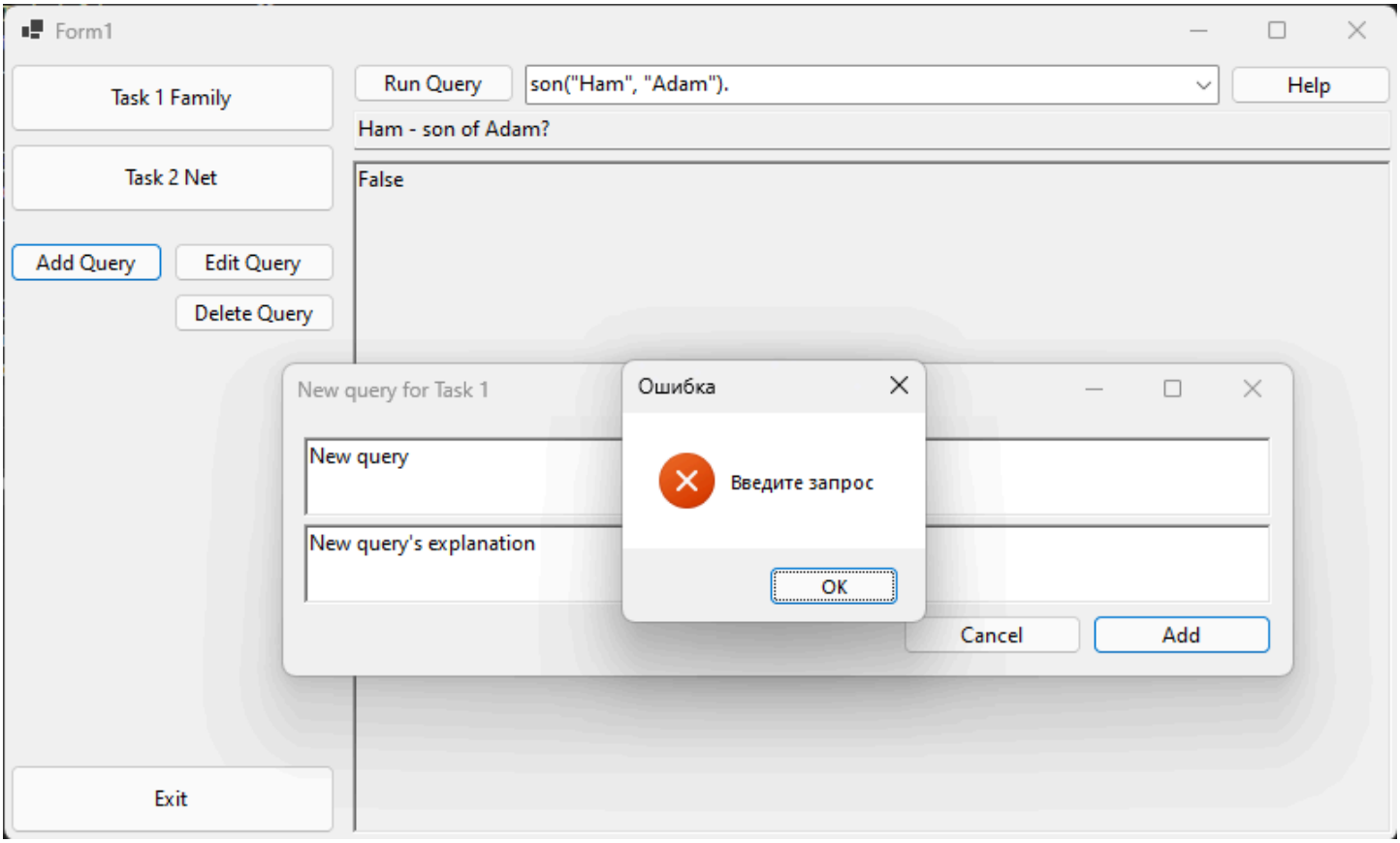


Рис. 8. Интерфейс приложения при попытке добавить пустой запрос.

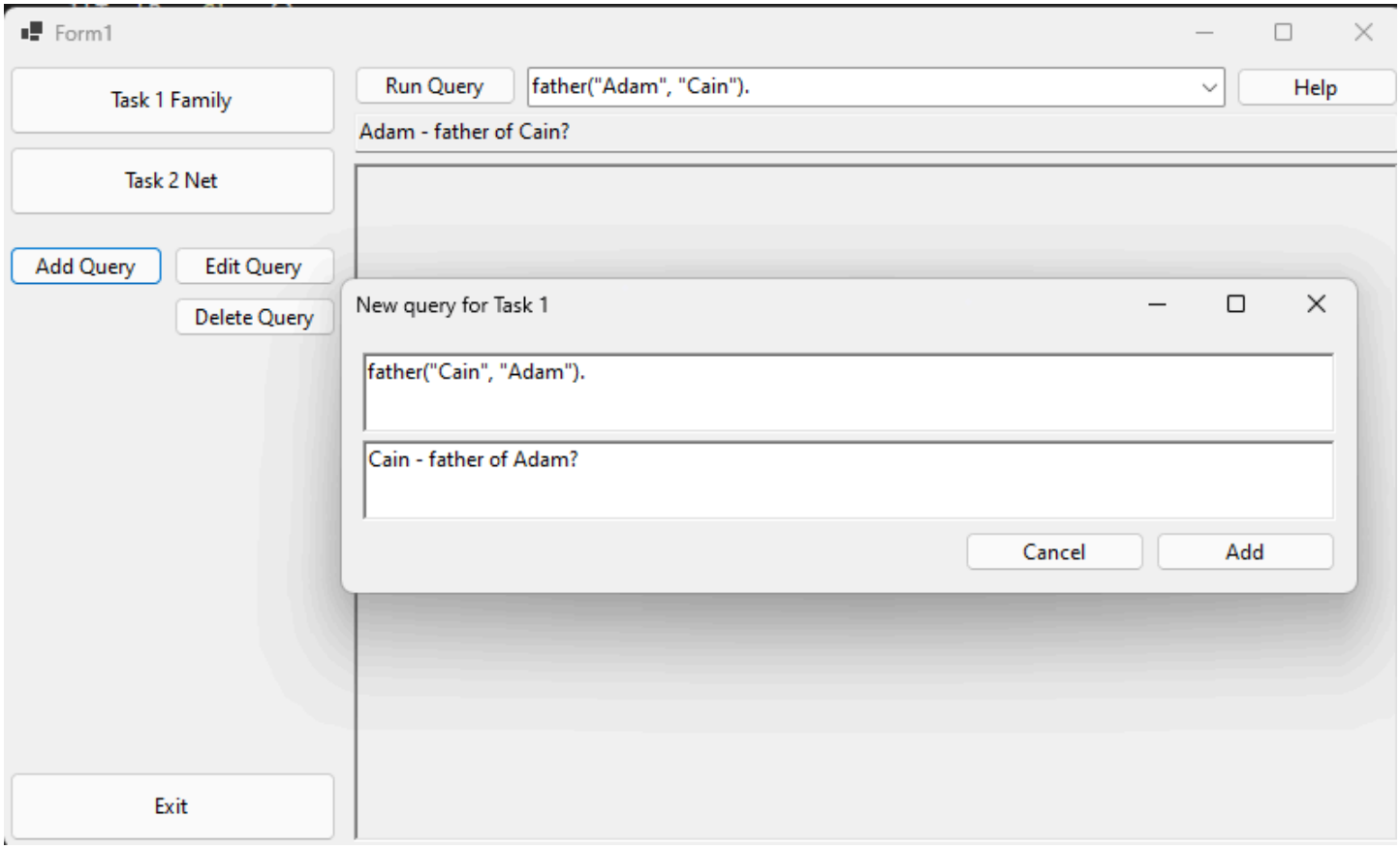


Рис. 9. Интерфейс приложения при вводе нового запроса.

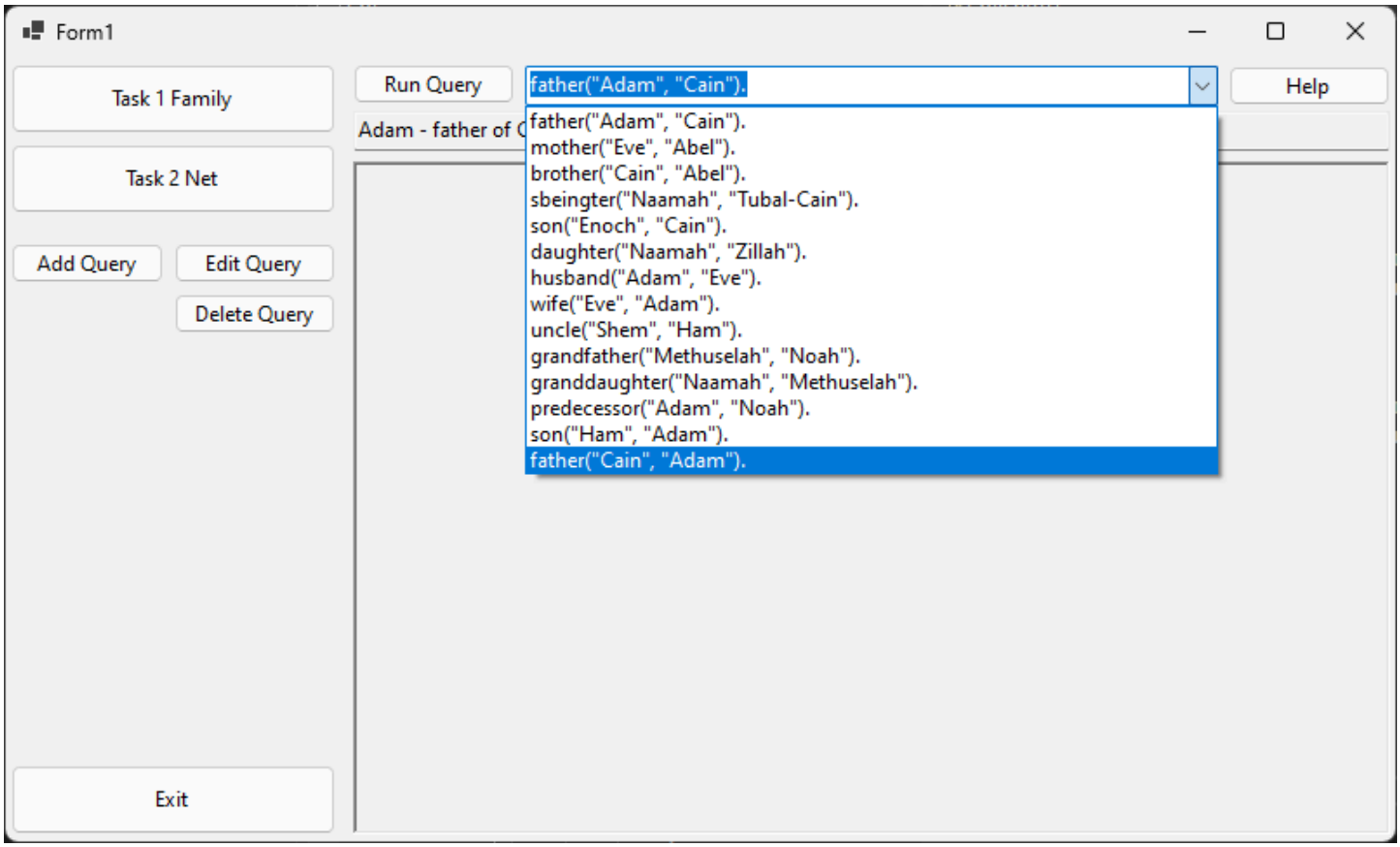


Рис. 10. Интерфейс приложения после добавления нового запроса.

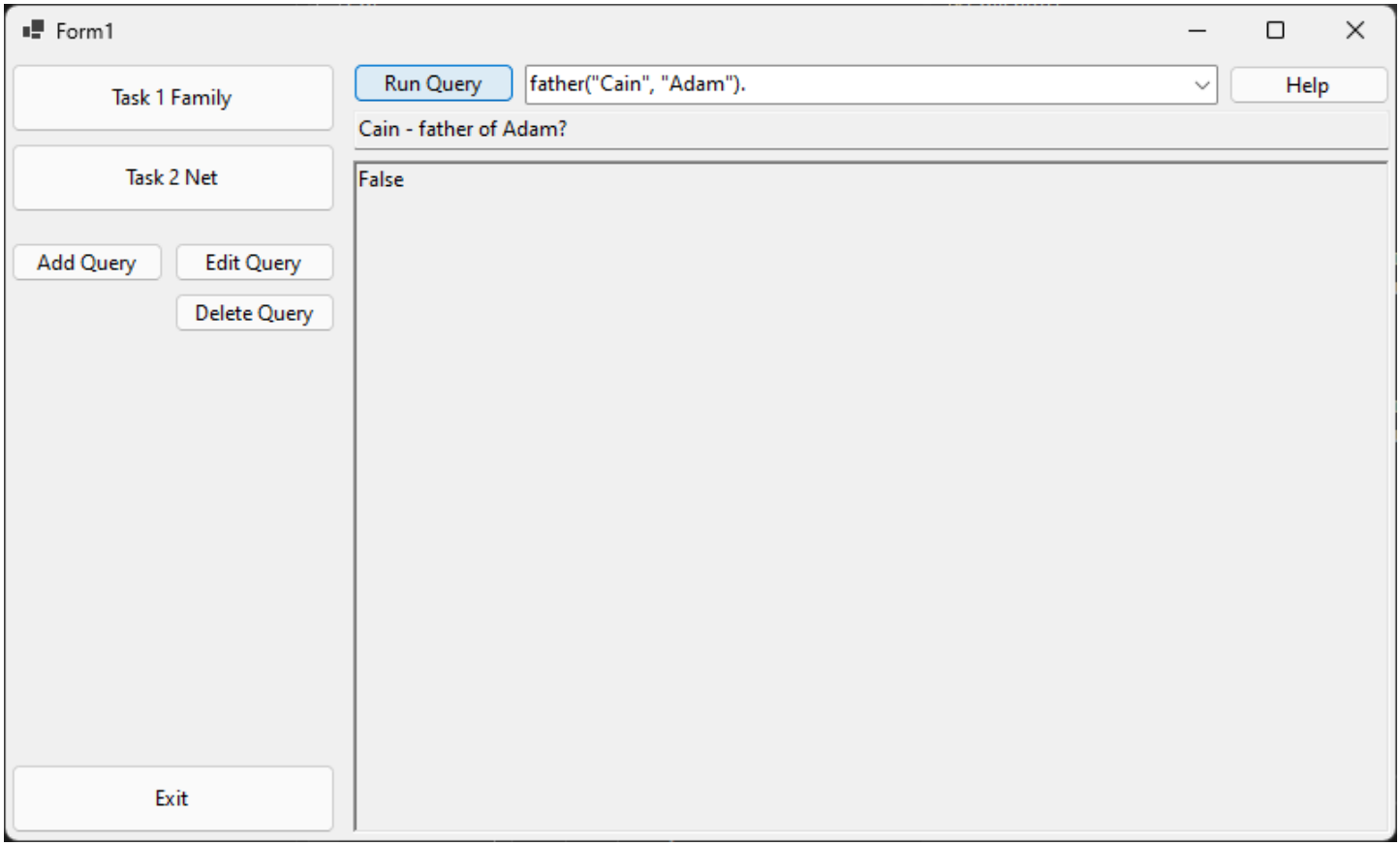


Рис. 11. Интерфейс приложения после запуска нового добавленного запроса.

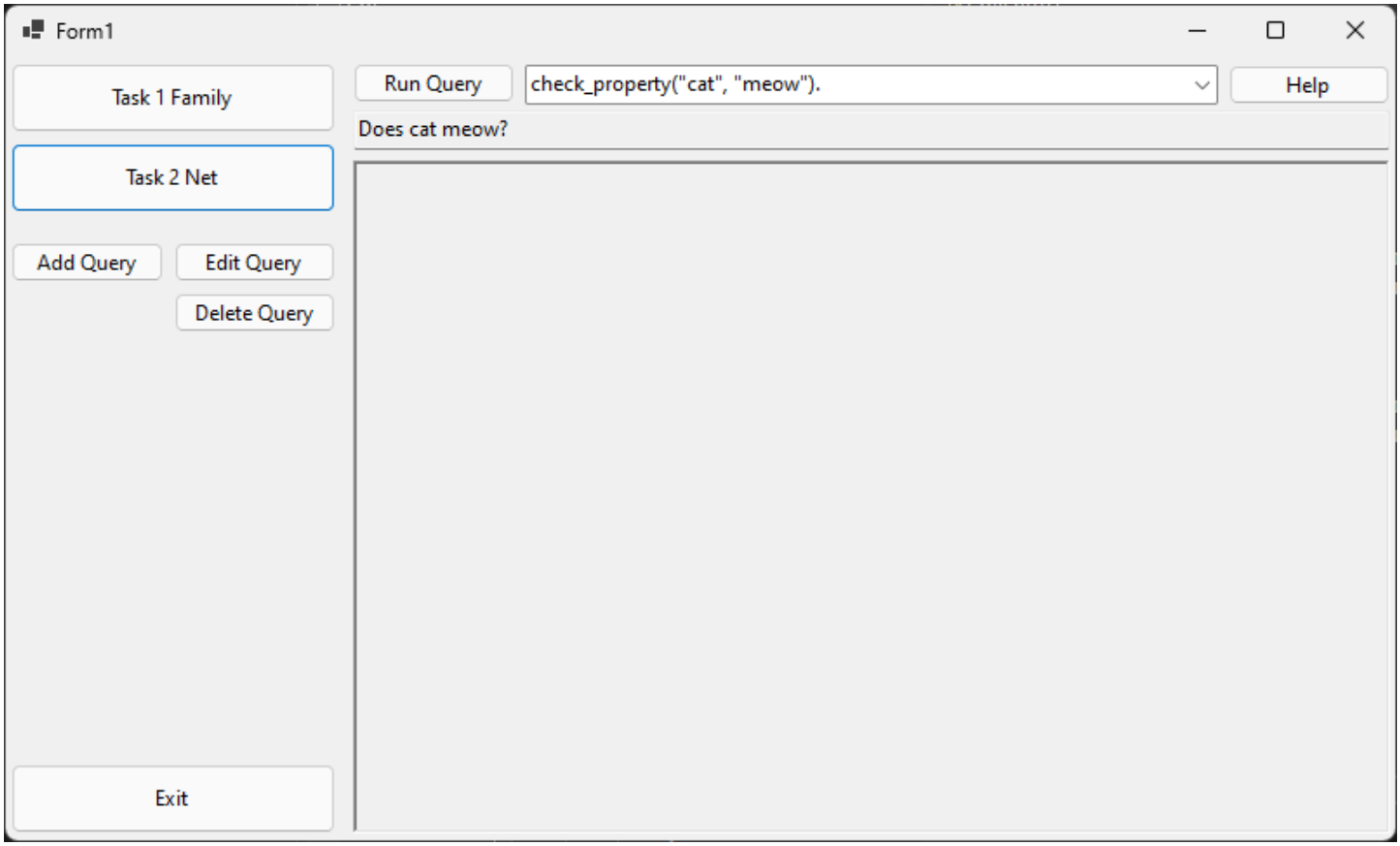


Рис. 12. Интерфейс приложения при выборе 2 задания.

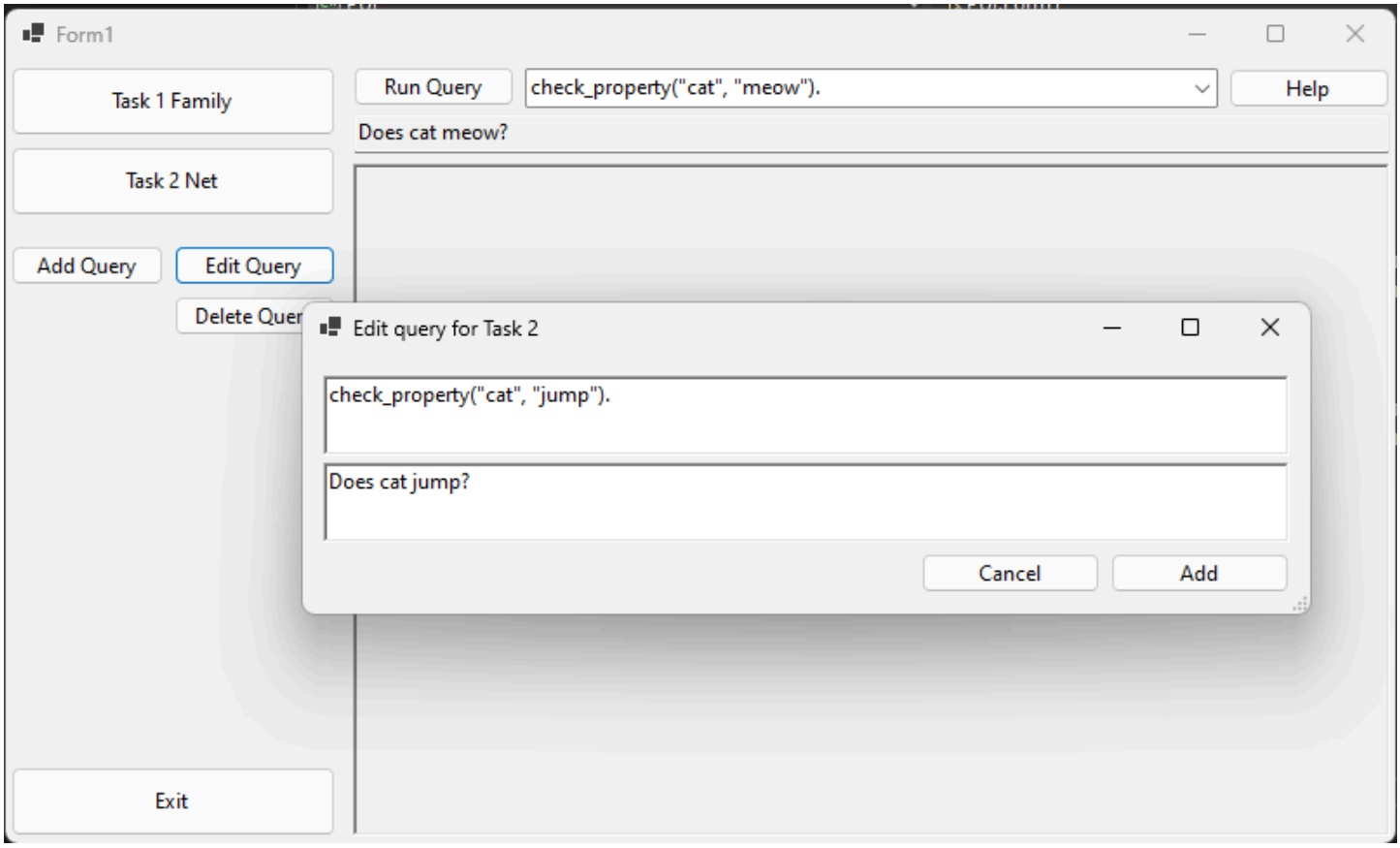


Рис. 13. Интерфейс приложения при изменении запроса.

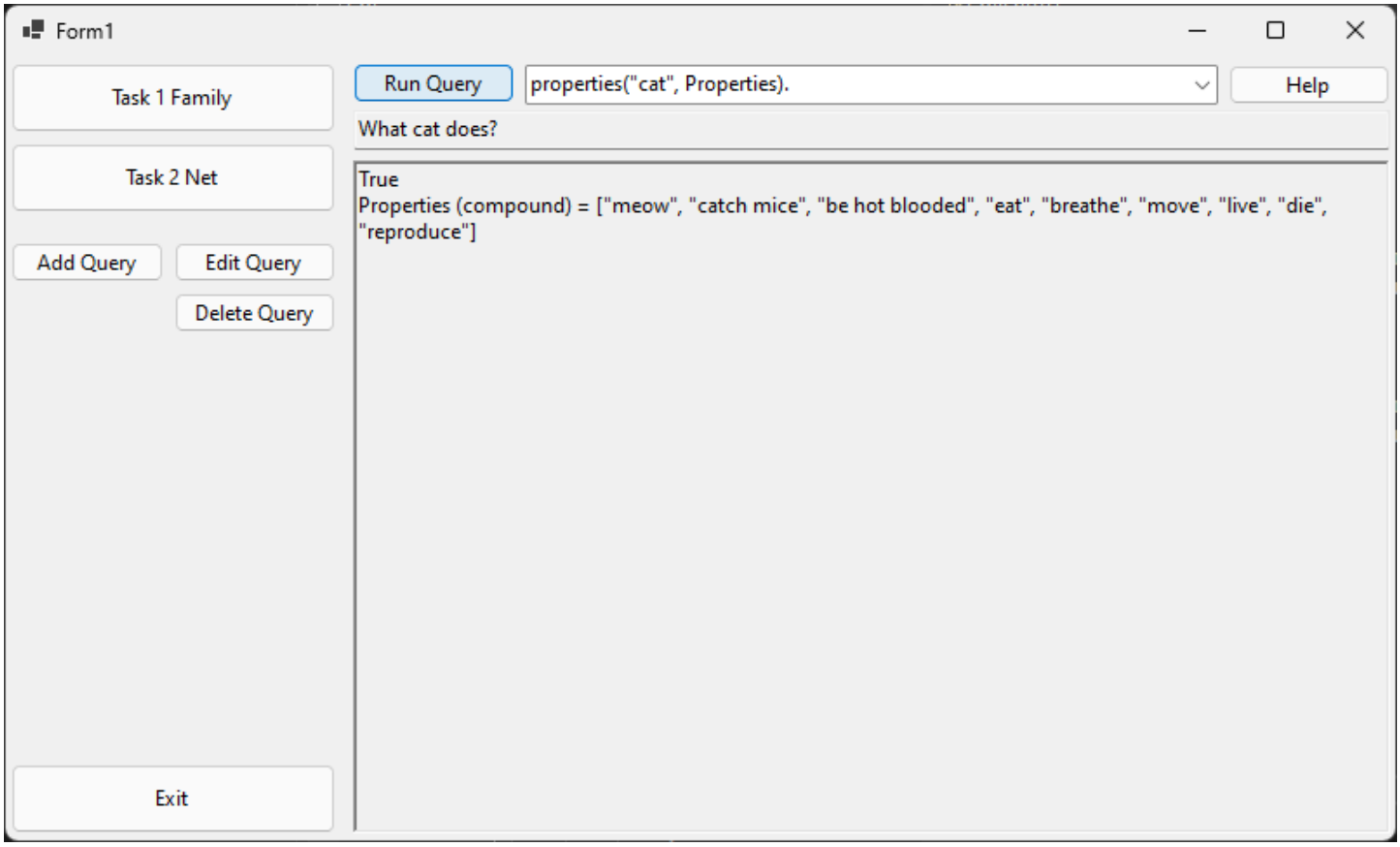


Рис. 14. Интерфейс приложения при запуске запроса на вывод списка.

## Вывод

В результате выполнения ЛР4 были разработаны семейное древо и семантическая сеть живых существ на языке SWI-Prolog и приложение с графическим интерфейсом для взаимодействия с разработанными программами на языке C#. Были реализованы рекурсивные правила, использованы встроенные возможности языка SWI-Prolog, графический интерфейс, возможность добавления, изменения и удаления запросов.

**Приложение А.**

**Листинг программы задания 1.**

```
% Факты о родительских отношениях
parent("Adam", "Cain").
parent("Eve", "Cain").
parent("Adam", "Abel").
parent("Eve", "Abel").
parent("Adam", "Seth").
parent("Eve", "Seth").

parent("Seth", "Enos").
parent("Enos", "Cainan").
parent("Cainan", "Mahalalel").
parent("Mahalalel", "Jared").
parent("Jared", "Enoch").
parent("Enoch", "Methuselah").
parent("Methuselah", "Lamech").
parent("Lamech", "Noah").
parent("Noah", "Shem").
parent("Noah", "Ham").
parent("Noah", "Japhet").

parent("Cain", "Enoch").
parent("Enoch", "Irad").
parent("Irad", "Mehujael").
parent("Mehujael", "Methushael").
parent("Methushael", "Lamech").

parent("Lamech", "Jabal").
parent("Adah", "Jabal").
parent("Lamech", "Jubal").
parent("Adah", "Jubal").

parent("Lamech", "Tubal-Cain").
parent("Zillah", "Tubal-Cain").
parent("Lamech", "Naamah").
parent("Zillah", "Naamah").

% Факты о женщинах
woman("Eve").
woman("Adah").
woman("Zillah").
woman("Naamah").

% Факты о мужчинах
man("Adam").
man("Seth").
man("Enos").
man("Cainan").
man("Mahalalel").
man("Jared").
man("Enoch").
man("Methuselah").
man("Lamech").
man("Noah").
man("Cain").
man("Irad").
man("Mehujael").
man("Methushael").
man("Abel").
```

```
man("Shem").
man("Ham").
man("Japhet").
man("Jabal").
man("Jubal").
man("Tubal-Cain").

% Факты о браке
married("Adam", "Eve").
married("Lamech", "Adah").
married("Lamech", "Zillah").

% Правила

% Проверка брака (в обе стороны)
check_married(X, Y) :-
    married(X, Y),
    married(Y, X).

% Отец и мать
father(X, Y) :-
    parent(X, Y),
    man(X).
mother(X, Y) :-
    parent(X, Y),
    woman(X).

% Брат и сестра
brother(X, Y) :-
    parent(Z, X),
    parent(Z, Y),
    man(X),
    X \= Y.
sister(X, Y) :-
    parent(Z, X),
    parent(Z, Y),
    woman(X),
    X \= Y.

% Сын и дочь
son(X, Y) :-
    parent(Y, X),
    man(X).
daughter(X, Y) :-
    parent(Y, X),
    woman(X).

% Муж и жена
husband(X, Y) :-
    check_married(X, Y),
    man(X).
wife(X, Y) :-
    check_married(X, Y),
    woman(X).

% Дядя и тетя
uncle(X, Y) :-
    parent(Z, Y),
```

```
    brother(X, Z).
aunt(X, Y) :-
    parent(Z, Y),
    sister(X, Z).

% Дедушка и бабушка
grandfather(X, Y) :-
    parent(X, Z),
    parent(Z, Y),
    man(X).
grandmother(X, Y) :-
    parent(X, Z),
    parent(Z, Y),
    woman(X).

% Внук и внучка
grandson(X, Y) :-
    parent(Z, Y),
    parent(X, Z),
    man(Y).
granddaughter(X, Y) :-
    parent(Z, Y),
    parent(X, Z),
    woman(Y).

% Предок (рекурсивное правило)
predecessor(X, Y) :-
    parent(X, Y).
predecessor(X, Y) :-
    parent(X, Z),
    predecessor(Z, Y).

% Примеры запросов:
% ?- father("Adam", "Cain").
% ?- mother("Eve", "Abel").
% ?- brother("Cain", "Abel").
% ?- sbeingter("Naamah", "Tubal-Cain").
% ?- son("Enoch", "Cain").
% ?- daughter("Naamah", "Zillah").
% ?- husband("Adam", "Eve").
% ?- wife("Eve", "Adam").
% ?- uncle("Shem", "Ham").
% ?- grandfather("Methuselah", "Noah").
% ?- granddaughter("Naamah", "Methuselah").
% ?- predecessor("Adam", "Noah").
```

**Приложение Б.**

**Листинг программы задания 2.**

```
% Факты наследования
being("animal", "creature").
being("fungi", "creature").
being("plant", "creature").

being("mammal", "animal").
being("fish", "animal").
being("amphibia", "animal").
being("reptile", "animal").
being("insect", "animal").
being("bird", "animal").

being("lower fungi", "fungi").
being("higher fungi", "fungi").

being("tree", "plant").
being("grass", "plant").
being("flower", "plant").

being("cat", "mammal").
being("cow", "mammal").
being("dolphin", "mammal").

being("shark", "fish").
being("salmon", "fish").

being("frog", "amphibia").

being("crocodile", "reptile").
being("turtle", "reptile").

being("ant", "insect").
being("bee", "insect").

being("sparrow", "bird").
being("penguin", "bird").

being("yeast", "lower fungi").

being("champignon", "higher fungi").
being("leccinum", "higher fungi").

being("oak", "tree").
being("birch", "tree").

being("wheat", "grass").
being("oats", "grass").

being("rose", "flower").
being("tulip", "flower").

% Факты свойств
does("creature", "live").
does("creature", "die").
does("creature", "reproduce").

does("animal", "eat").
does("animal", "breathe").
```

```
does("animal", "move").

does("fungi", "not breathe").
does("fungi", "not move").

does("plant", "photosynthesize").
does("plant", "not move").

does("mammal", "be hot blooded").
does("fish", "live in water").
does("amphibia", "be able to live in water").
does("amphibia", "be able to live on land").
does("reptile", "be cold blooded").
does("insect", "have six legs").
does("bird", "can fly").
does("bird", "lay eggs").

does("lower fungi", "be parasite").
does("higher fungi", "be edible").

does("tree", "give shade").
does("grass", "feed animals").
does("flower", "smell").

does("cat", "meow").
does("cat", "catch mice").

does("cow", "moo").
does("cow", "give milk").

does("dolphin", "live in water").

does("shark", "be predator").
does("salmon", "migrate").

does("frog", "jump").

does("crocodile", "be predator").
does("turtle", "live long").

does("ant", "be worker").
does("bee", "produce honey").

does("sparrow", "be small").
does("penguin", "not fly").

does("yeast", "ferment").

does("champignon", "be short").
does("leccinum", "be red").

does("oak", "live long").
does("birch", "be black and white").

does("wheat", "be source of bread").
does("oats", "be healthy").

does("rose", "be spiky").
```

```
does("tulip", "grow from bulb").

% Правила наследования
inherit(X, Z) :-
    being(X, Y),
    does_too(Y, Z).

does_too(X, Z) :-
    does(X, Z);
    inherit(X, Z).

ancestor(X, Y) :-
    being(X, Y).
ancestor(X, Y) :-
    being(X, Z),
    ancestor(Z, Y).

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

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

ancestors(Y, Ancestors) :-
    findall(Ancestor, ancestor(Ancestor, Y), Ancestors).

% Примеры использования:
% ?- check_property("cat", "meow").
% ?- check_property("cat", "breathe").
% ?- properties("cat", Properties).
% ?- check_property("dolphin", "eat").
% ?- ancestors("animal", Ancestors).
% ?- ancestors("fungi", Ancestors).
% ?- ancestors("creature", Ancestors).
```

## Приложение В.

## Листинг программы задания 3.

**Form1.cs**

```
using Prolog;

namespace PUI
{
    public partial class Form1 : Form
    {
        PrologEngine prolog = new(persistentCommandHistory: false);
            string task1 = "E:\\SAVVA\\STUDY\\MPEI\\10_sem\\TPПO\\PUI\\PUI\\task1-
family.pl";
            string task2 = "E:\\SAVVA\\STUDY\\MPEI\\10_sem\\TPПO\\PUI\\PUI\\task2-
net.pl";

        string[] queries1 = new string[]
        {
                        "father(\"Adam\", \"Cain\").",
                        "mother(\"Eve\", \"Abel\").",
                        "brother(\"Cain\", \"Abel\").",
                        "sbeingter(\"Naamah\", \"Tubal-Cain\").",
                        "son(\"Enoch\", \"Cain\").",
                        "daughter(\"Naamah\", \"Zillah\").",
                        "husband(\"Adam\", \"Eve\").",
                        "wife(\"Eve\", \"Adam\").",
                        "uncle(\"Shem\", \"Ham\").",
                        "grandfather(\"Methuselah\", \"Noah\").",
                        "granddaughter(\"Naamah\", \"Methuselah\").",
                        "predecessor(\"Adam\", \"Noah\").",
                        "son(\"Ham\", \"Adam\").",
        };
        string[] explanations1 = new string[]
        {
                        "Adam - father of Cain?",
                        "Eve - mother of Abel?",
                        "Cain - brother of Abel?",
                        "Naamah - sister of Tubal-Cain?",
                        "Enoch - son of Cain?",
                        "Naamah - daughter of Zillah?",
                        "Adam - husband of Eve?",
                        "Eve - wife of Adam?",
                        "Shem - uncle of Ham?",
                        "Methuselah - grandfather of Noah?",
                        "Naamah - granddaughter of Methuselah?",
                        "Adam - predecessor of Noah?",
                        "Ham - son of Adam?"
        };
        string[] queries2 = new string[]
        {
                        "check_property(\"cat\", \"meow\").",
                        "check_property(\"cat\", \"breathe\").",
                        "properties(\"cat\", Properties).",
                        "check_property(\"dolphin\", \"eat\").",
                        "ancestors(\"animal\", Ancestors).",
                        "ancestors(\"fungi\", Ancestors).",
                        "ancestors(\"creature\", Ancestors)."
        };
        string[] explanations2 = new string[]
        {
```

```
                        "Does cat meow?",
                        "Does cat breathe?",
                        "What cat does?",
                        "Does dolphin eat?",
                        "Animals list",
                        "Fungis list",
                        "Creatures list"
            };
            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;
                queryTextBox.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;
                queryTextBox.Clear();
                queryComboBox.Items.Clear();
                queryComboBox.Items.AddRange(queries1);
                queryComboBox.SelectedIndex = 0;
                      solutions = prolog.GetAllSolutions(sourceFileName: task1, query:
queries1[queryComboBox.SelectedIndex]);
                queryTextBox.Text = explanations1[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)
        {
            queryTextBox.Clear();
            if (isTask1)
            {
                    solutions = prolog.GetAllSolutions(sourceFileName: task1, query:
queries1[queryComboBox.SelectedIndex]);
                queryTextBox.Text = explanations1[queryComboBox.SelectedIndex];
            }
            else
            {
                    solutions = prolog.GetAllSolutions(sourceFileName: task2, query:
queries2[queryComboBox.SelectedIndex]);
                queryTextBox.Text = explanations2[queryComboBox.SelectedIndex];
            }

            resultTextBox.Clear();
        }
        private void task2Button_Click(object sender, EventArgs e)
        {
            queryComboBox.Enabled = true;
            queryTextBox.Clear();
            queryComboBox.Items.Clear();
            queryComboBox.Items.AddRange(queries2);
            queryComboBox.SelectedIndex = 0;
                  solutions = prolog.GetAllSolutions(sourceFileName: task2, query:
queries2[queryComboBox.SelectedIndex]);
            queryTextBox.Text = explanations2[queryComboBox.SelectedIndex];
            addButton.Enabled = true;
            editButton.Enabled = true;
            deleteButton.Enabled = true;
            runButton.Enabled = true;
            isTask1 = false;
        }

        private void exitButton_Click(object sender, EventArgs e)
        {
            Application.Exit();
        }

        private void addButton_Click(object sender, EventArgs e)
        {
            Form2 form2 = new Form2();
            form2.Text = "New query for Task";
            if (isTask1)
            {
                form2.Text += " 1";
            }
            else
            {
                form2.Text += " 2";
```

```
                }
                form2.ShowDialog();
                if (form2.DialogResult == DialogResult.OK)
                {
                    if (isTask1)
                    {
                        queries1 = queries1.Append(form2.newQuery).ToArray();
                                                                explanations1 =
explanations1.Append(form2.newQueryExplanation).ToArray();
                    }
                    else
                    {
                        queries2 = queries2.Append(form2.newQuery).ToArray();
                                                                explanations2 =
explanations2.Append(form2.newQueryExplanation).ToArray();
                    }
                    queryComboBox.Items.Clear();
                    queryTextBox.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 for Task";
                if (isTask1)
                {
                    form3.Text += " 1";
                }
                else
                {
                    form3.Text += " 2";
                }
                if (isTask1)
                {
                    form3.editedQuery = queries1[queryComboBox.SelectedIndex];
                                            form3.editedQueryExplanation =
explanations1[queryComboBox.SelectedIndex];
                }
                else
                {
                    form3.editedQuery = queries2[queryComboBox.SelectedIndex];
                                            form3.editedQueryExplanation =
explanations2[queryComboBox.SelectedIndex];
                }
                form3.ShowDialog();
                if (form3.DialogResult == DialogResult.OK)
                {
```

```
                    if (isTask1)
                    {
                        queries1[queryComboBox.SelectedIndex] = form3.editedQuery;
                                        explanations1[queryComboBox.SelectedIndex] =
form3.editedQueryExplanation;
                        queryComboBox.Items.Clear();
                        queryTextBox.Clear();
                        queryComboBox.Items.AddRange(queries1);
                    }
                    else
                    {
                        queries2[queryComboBox.SelectedIndex] = form3.editedQuery;
                                        explanations2[queryComboBox.SelectedIndex] =
form3.editedQueryExplanation;
                        queryComboBox.Items.Clear();
                        queryTextBox.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();
                        explanations1 = explanations1.Where((source, index) => index !=
queryComboBox.SelectedIndex).ToArray();
                    queryComboBox.Items.Clear();
                    queryTextBox.Clear();
                    queryComboBox.Items.AddRange(queries1);
                }
                else
                {
                                queries2 = queries2.Where((source, index) => index !=
queryComboBox.SelectedIndex).ToArray();
                        explanations2 = explanations2.Where((source, index) => index !=
queryComboBox.SelectedIndex).ToArray();
                    queryComboBox.Items.Clear();
                    queryTextBox.Clear();
                    queryComboBox.Items.AddRange(queries2);
                }
                queryComboBox.SelectedIndex = 0;
                queryComboBox_SelectedIndexChanged(sender, e);
            }
        }
    }
```

**Form2.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
```

```
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PUI
{
    public partial class Form2 : Form
    {
        public Form2()
        {
            InitializeComponent();
            newQueryTextBox.Text = "New query";
            explanationTextBox.Text = "New query's explanation";
        }

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

        private void addButton_Click(object sender, EventArgs e)
        {
            if (newQueryTextBox.Text == "New query")
            {
                    MessageBox.Show("Введите запрос", "Ошибка", MessageBoxButtons.OK,
MessageBoxIcon.Error);
            }
            if (explanationTextBox.Text == "New query's explanation")
            {
                        MessageBox.Show("Введите объяснение запроса", "Ошибка",
MessageBoxButtons.OK, MessageBoxIcon.Error);
            }
            newQuery = newQueryTextBox.Text;
            newQueryExplanation = explanationTextBox.Text;
        }

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

**Form3.cs**

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace PUI
{
    public partial class Form3 : Form
    {
```

```
            public string editedQuery { get; set; } = null;
            public string editedQueryExplanation { get; set; } = null;
            public Form3()
            {
                InitializeComponent();
                editQueryTextBox.Text = editedQuery;
                explanationTextBox.Text = editedQueryExplanation;
            }

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

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