**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра математического обеспечения и применения ЭВМ**

отчет

**по лабораторной работе №2**

**по дисциплине «Базы знаний и экспертные системы»**

**Тема: Разработка и отладка простой программы на языке Visual Prolog**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6381 |  | Шевелева А.М. |
| Преподаватель |  | Сучков А.И. |

Санкт-Петербург

2019

Цель работы.

Получение необходимых навыков для разработки и отладки простейшей программы на языке Visual Prolog.

Основные теоретические положения

Prolog является декларативным языком программирования: логика программы выражается в терминах отношений, представленных в виде фактов и правил. Для того чтобы инициировать вычисления, выполняется специальный запрос к базе знаний, на которые система логического программирования генерирует ответы «истина» и «ложь». Для обобщённых запросов с переменными в качестве аргументов созданная система Пролог выводит конкретные данные в подтверждение истинности обобщённых сведений и правил вывода.

Иначе говоря, предикат можно определить как функцию, отображающую множество произвольной природы в множество булевых значений {ложно, истинно}. Задача пролог-программы заключается в том, чтобы доказать, является ли заданное целевое утверждение следствием из имеющихся фактов и правил.

Язык программирования, реализованный в Visual (а ранее в Turbo) Prolog'е отличается от классического пролога тем, что он основан на строгой статической типизации. В него также добавлены средства объектно-ориентированного программирования, анонимные предикаты (лямбда-предикаты), факты-переменные и разрушающее присваивание для них, аргументы-домены (Generic Interfaces and Classes) и параметрический полиморфизм, мониторы (Monitors with guards), императивные конструкции (foreach, if...then...else), коллекторы списков ([...||...]) и пр.

Постановка задачи.

Разработать программу на языке Visual Prolog, описывающие родственные связи между собаками, а также реализовать генеалогическое древо Вашей семьи. Порядок выполнения следующий:

1. Установить в опциях среды шрифт с русской раскладкой.
2. Установить опции компилятора.
3. Создать новый проект DOG с заданным кодом программы.
4. Запустить отладчик и проанализировать трассу выполнения программы.
5. Написать программу DOG1, описывающую родственные связи собак в соответствии с вариантом задания.
6. Произвести отладку программы на данных в задании запросах и построить для каждого запроса трассу выполнения программы.
7. Написать программу FAMILY, описывающую родственные связи своей семьи и содержащую не менее 8 правил вывода родственных отношений в семье, и отладить программу на 5-6 различных запросах.

Выполнение работы.

Установлен в опциях среды шрифт Courier с раскладкой Кириллица **на карточке Fonts (Options/Global/Environment) окна редактора** Editor Windows.

Установлены опции компилятора (Options/Project/Compiler Options): на карточке Output установлены Generate Debug Information, Disable Optimizations, на карточке Warnings установлен Nondeterm, и сняты Unused Variables и Unused Predicates. Результат работы представлен на рис. 1-2.

|  |  |
| --- | --- |
|  |  |
| Рисунок 1 − Изменения в карточке Output | Рисунок 2 − Изменения в карточке Warnings |

Создан новый проект DOG. В созданном проекте удален модуль VPITOOL.PRO (см. рис. 3). Из модуля DOG.PRO удалено все содержимое и вставлен текст программы, представленный на рис. 4.

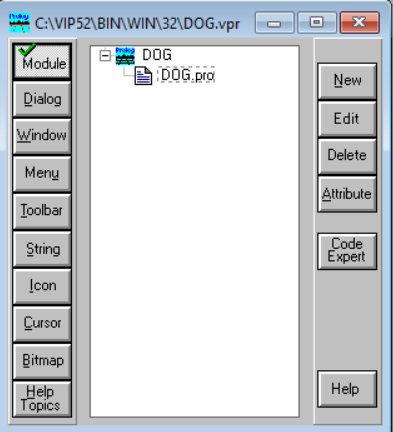


Рисунок 3 − Новый проект DOG

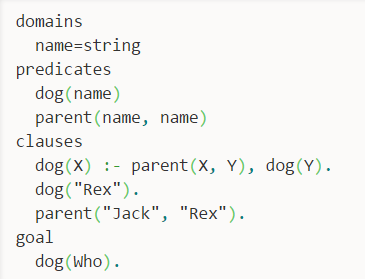


Рисунок 4 − Код программы DOG

Запущена программа на выполнение через кнопку Test Goal на панели инструментов. **Результат представлен на рис. 5**.

В программе были созданы два факта: что Rex является собакой и что Jack является родителем Rex. А также было создано правило, которое говорит, что если X и Y являются родителем и ребенком и Y − собака, то Х тоже является собакой. Целью программы был вывод всех собак. При запуске программы сначала было проверено правило и определено, что Jack является собакой. Потом в вывод был добавлен факт, что Rex также является собакой.

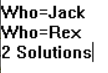


Рисунок 5 − Результат работы программы DOG

**Выполнена компиляция, построение и запущен отладчик. В среде отладчика включены окна просмотра переменных и стека вызова (см. рис. 6). Произведена пошаговая прокрутка программы. Трасса выполнения программы (см. приложение А)** подтверждает ранее проанализированный результат.

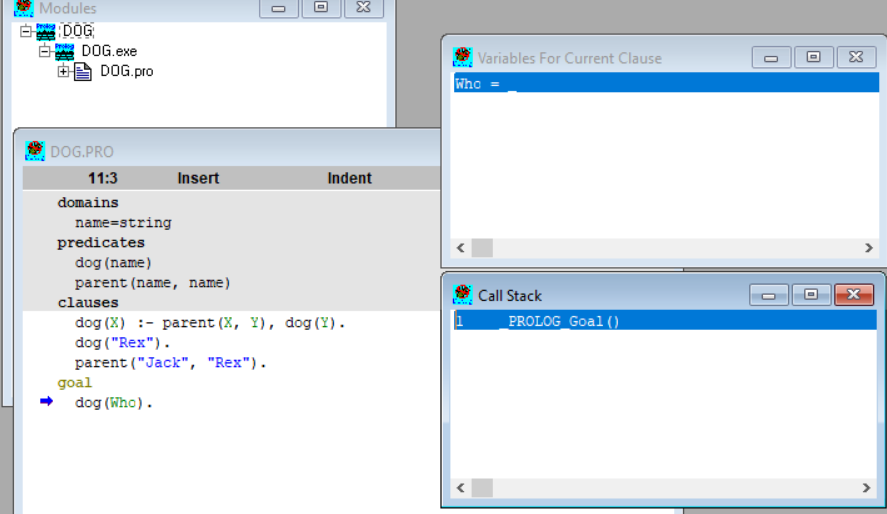


Рисунок 6 − Среда отладчика. **Окна просмотра переменных и стека вызова**

Создана программа DOG1, описывающая родственные связи собак в соответствии с рис. 7. Она содержит предикаты dog и parent и 2 правила. Полный код программы приведен в приложении Б.

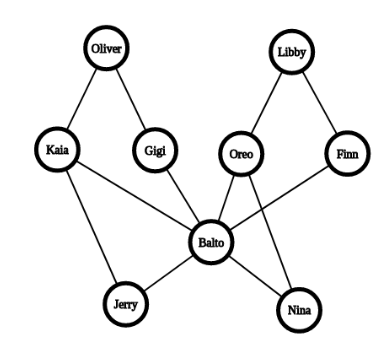


Рисунок 7 − Граф родственных связей собак

Произведена отладка программы на следующих запросах:

Кто является собакой?

Кто является родителем?

Кто является внуком (внучкой)?

Balto – собака?

Кто родитель собаки Balto?

Кому Balto является родителем?

Ответ на первый запрос получается из правила родства (dog(X)) и фактов. На второй вопрос ответ получается из фактов родства. На третий запрос ответ выводится из правила grandchild(X) и фактов. Ответ на 4-6 вопросы получается из фактов родства. Результаты выполнения запросов представлены на рис. 8-13.

Трассы выполнения запросов представлены в приложении В.

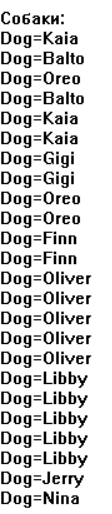


Рисунок 8 − Результат запроса «Кто является собакой?»



Рисунок 9 − Результат запроса «Кто является родителем?»



Рисунок 10 − Результат запроса «Кто является внуком (внучкой)?»



Рисунок 11 − Результат запроса «Balto – собака?»

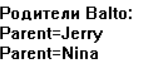


Рисунок 12 − Результат запроса «Кто родитель собаки Balto?»



Рисунок 13 − Результат запроса «Кому Balto является родителем?»

Написана на языке Visual Prolog программа *FAMILY*, содержащая 12 правил вывода для родственных отношений в семье (мать, отец, сестра, брат, племянница, племянник, внучка/внук, бабушка/дедушка, двоюродная сестра, двоюродный брат и зять(муж сестры)). Дерево родственных связей представлено на рис. 14. Использованы предикаты parent(symbol, symbol), man(symbol), woman(symbol), married(symbol, symbol), и записаны факты, описывающие семью.  Полный код программы представлен в приложении Г.

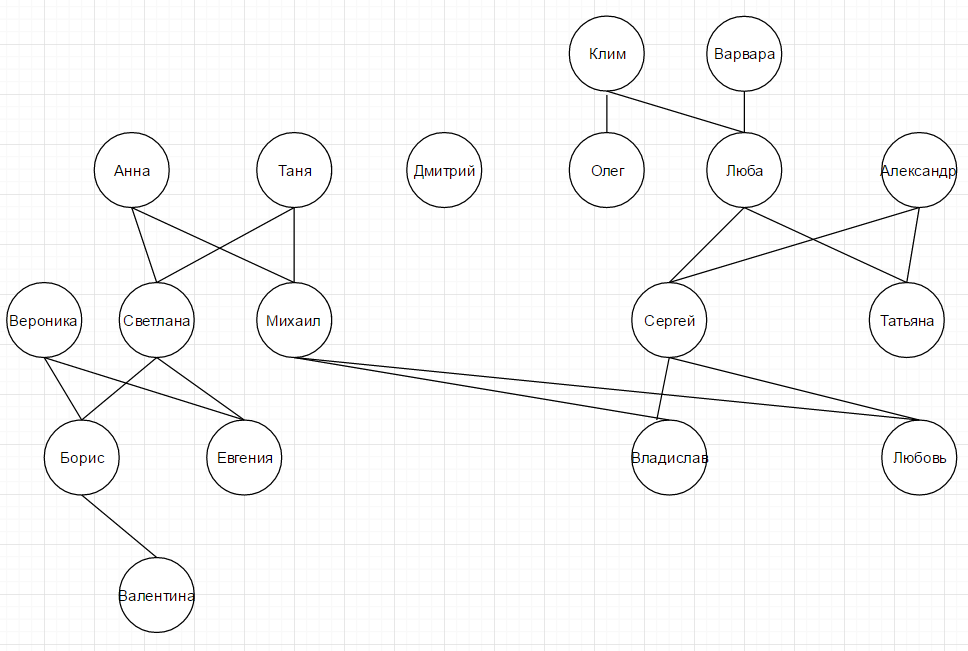


Рисунок 14 − Дерево родственных связей семьи

Отладка программы:

Запрос «Вывод списка дедушек в семье». Ответ по дереву семьи: Сергей, Владислав и Борис. Результат (см. рис. 15) программы совпал с ответом.

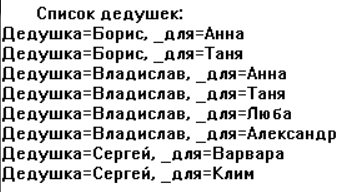


Рисунок 15 − Запрос «Вывод списка дедушек в семье»

Запрос «Вывод списка мам в семье». Ответ по дереву семьи: Валентина, Евгения, Светлана, Любовь, Татьяна, Люба. Результат (см. рис. 16) программы совпал с ответом.

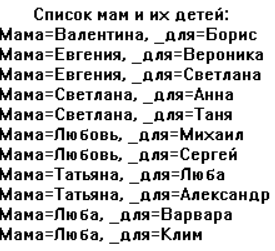


Рисунок 16 − Запрос «Вывод списка мам в семье»

Запрос «Вывод списка пап в семье». Ответ по дереву семьи: Борис, Михаил, Владислав, Сергей, Олег. Результат (см. рис. 17) программы совпал с ответом.



Рисунок 17 − Запрос «Вывод списка пап в семье»

Запрос «Вывод списка сестер в семье». Ответ по дереву семьи: Вероника и Светлана, Анна и Татьяна, Люба для Александра, Клим для Варвары. Результат (см. рис. 18) программы совпал с ответом.

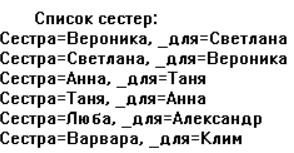


Рисунок 18 − Запрос «Вывод списка сестер в семье»

Запрос «Вывод списка братьев в семье». Ответ по дереву семьи: Михаил и Сергей, Александр для Любы, Клим для Варвары. Результат (см. рис. 19) программы совпал с ответом.

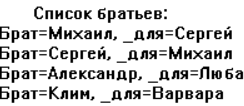


Рисунок 19 − Запрос «Вывод списка братьев в семье»

Запрос «Вывод списка двоюродных сестер в семье». Ответ по дереву семьи: Анна и Люба, Таня и Люба, Анна и Таня для Александра. Результат (см. рис. 20) программы совпал с ответом.

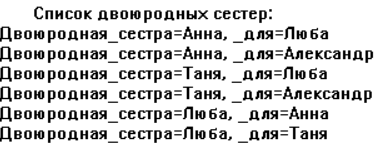


Рисунок 20 − Запрос «Вывод списка двоюродных сестер в семье»

Запрос «Вывод списка племянников в семье». Ответ по дереву семьи: Клим для Александра и Александр для Михаила. Результат (см. рис. 21) программы совпал с ответом.

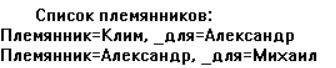


Рисунок 21 − Запрос «Вывод списка племянников в семье»

Запрос «Вывод списка племянниц в семье». Ответ по дереву семьи: Анна и Таня для Вероники, Варвара для Александра, Анна и Таня для Сергея, Люба для Михаила. Результат (см. рис. 22) программы совпал с ответом.

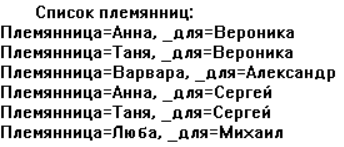


Рисунок 22 − Запрос «Вывод списка племянниц в семье»

Запрос «Вывод списка зятей (муж сестры) в семье». Ответ по дереву семьи: Михаил, Дмитрий и Олег. Результат (см. рис. 23) программы совпал с ответом.

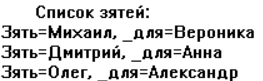


Рисунок 23 − Запрос «Вывод списка зятей (муж сестры) в семье»

Выводы.

Была разработана программа на языке Visual Prolog, описывающая родственные связи между собаками в соответствии с данным в задании графом. Программа была протестирована на 6 заданных запросах. По каждому запросу был получен корректный ответ в соответствии с графом родства. Также для каждого запроса была создана трасса выполнения программы, используя данные из отладчика.

Была разработана программа на языке Visual Prolog, описывающая родственные связи между членами семьи. Для этого было создано генеалогическое дерево семьи. По генеалогическому дереву были выявлены факты родства. Для определения некоторых родственных связей были написаны правила родства. Программа была протестирована на девяти запросах. Каждый запрос выдал верный в соответствии с генеалогическим деревом ответ.

**ПРИЛОЖЕНИЕ А**

**ТРАССА ВЫПОЛНЕНИЯ ОДНОГО РЕШЕНИЯ ПРОГРАММЫ РОДСТВЕННЫХ СВЯЗЕЙ СОБАК**

**Трасса выполнения одного решения программы DOG представлена в табл.1.**

**Таблица 1** − **Трасса выполнения одного решения программы DOG**

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 1 | Dog = \_ | 1 \_PROLOG\_Goal() | Просмотр цели prolog-программы. |
| 2 | - | 1 dog( \_ )  2 \_PROLOG\_Goal() | Добавление цели в Call Stack. |
| 3 | X = \_  Y = \_ | 1 dog( \_ )  2 \_PROLOG\_Goal() | Просмотр правила родства. Имеем две переменные X, Y, которые нужно определить. |
| 4 | - | 1 parent( \_, \_ )  2 dog( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack правила родства. |
| 5 | X = "Jack"  Y = "Rex" | 1 dog( \_ )  2 \_PROLOG\_Goal() | Нахождение факта, что Jack является родителем Rex. X=Jack, Y=Rex. |
| 6 | X = "Rex"  Y = \_ | 1 dog( "Rex" )  2 dog( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения, что Rex - собака. Подстановка в правило родства X=Rex, Y=неизвестно |

Окончание таблицы 1

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение |
| 7 | - | 1 parent( \_, "Rex" )  2 dog( "Rex" )  3 dog( \_ )  4 \_PROLOG\_Goal() | Добавление в Call Stack утверждения, где родителем является Rex, а его ребенок неизвестен. Проверка утверждения. |
| 8 | - | 1 dog( "Rex" )  2 dog( \_ )  3 \_PROLOG\_Goal() | Утверждение неверно, оно удаляется из Call Stack. Проверка утверждения, что Rex - собака. |
| 9 | X = "Jack"  Y = "Rex" | 1 dog( \_ )  2 \_PROLOG\_Goal() | Утверждение верно, оно удаляется из Call Stack. Утверждение родства X=Kaia, Y=Jerry верно. |
| 10 | Dog = "Jack" | 1 dog( \_ )  2 \_PROLOG\_Goal() | Одно решение цели найдено |

**ПРИЛОЖЕНИЕ Б**

**ПРОГРАММА ОПИСАНИЯ РОДСТВЕННЫХ СВЯЗЕЙ СОБАК**

domains

name=string

predicates

dog(name)

parent(name, name)

grandchild(name)

clauses

grandchild(X) :- parent(Y, X), parent(Z, Y).

dog(X) :- parent(Y, X), dog(Y).

dog("Jerry").

dog("Nina").

parent("Jerry", "Kaia").

parent("Jerry", "Balto").

parent("Nina", "Oreo").

parent("Nina", "Balto").

parent("Balto", "Kaia").

parent("Balto", "Gigi").

parent("Balto", "Oreo").

parent("Balto", "Finn").

parent("Kaia", "Oliver").

parent("Gigi", "Oliver").

parent("Oreo", "Libby").

parent("Finn", "Libby").

goal

nl,write("Собаки: "),nl,dog(Dog);

nl,write("Родители: "),nl,parent(Parent,\_);

nl,write("Внуки: "),nl,grandchild(Grandchild);

nl,write("Родители Balto: "),nl,parent(Parent, "Balto");

nl,write("Дети Balto: "),nl,parent("Balto", Child);

nl,write("Balto собака? Ответ: "), dog("Balto").

**ПРИЛОЖЕНИЕ В**

**ТРАССЫ ВЫПОЛНЕНИЯ ЗАПРОСОВ ПРОГРАММЫ РОДСТВЕННЫХ СВЯЗЕЙ СОБАК**

Трассы выполнения запросов, заданных в задании, представлены в табл. 2-7.

Таблица 2 − Трасса выполнения запроса «Кто является собакой?»

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 1 | Dog = \_ | 1 \_PROLOG\_Goal() | Просмотр цели prolog-программы |
| 2 | - | 1 dog( \_ )  2 \_PROLOG\_Goal() | Добавление цели в Call Stack |
| 3 | X = \_  Y = \_ | 1 dog( \_ )  2 \_PROLOG\_Goal() | Просмотр второго правила программы (правило родства). Имеем две переменные X, Y, которые нужно определить |
| 4 | - | 1 parent( \_, \_ )  2 dog( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack правила родства. |
| 5 | X = "Kaia"  Y = "Jerry" | 1 dog( \_ )  2 \_PROLOG\_Goal() | Нахождение 1 факта родства. X=Kaia, Y=Jerry. |
| 6 | X = "Jerry"  Y = \_ | 1 dog( "Jerry" )  2 dog( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения, что Jerry - собака. Подстановка в правило родства X=Jerry, Y=неизвестно. |

Окончание таблицы 2

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменные | Стек вызовов | Пояснения работы |
| 7 | - | 1 parent( \_, "Jerry" )  2 dog( "Jerry" )  3 dog( \_ )  4 \_PROLOG\_Goal() | Добавление в Call Stack утверждения, где ребенком является Jerry, а его родитель неизвестен. Проверка утверждения. |
| 8 | - | 1 dog( "Jerry" )  2 dog( \_ )  3 \_PROLOG\_Goal() | Утверждение неверно, оно удаляется из Call Stack. Проверка утверждения, что Jerry - собака. |
| 9 | X = "Kaia"  Y = "Jerry" | 1 dog( \_ )  2 \_PROLOG\_Goal() | Утверждение верно, оно удаляется из Call Stack. Утверждение родства X=Kaia, Y=Jerry верно. |
| 10 | Dog = "Kaia" | 1 dog( \_ )  2 \_PROLOG\_Goal() | Одно решение цели найдено. |

Таблица 3 − Трасса выполнения запроса «Кто является родителем?»

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 1 | Parent = \_ | 1 \_PROLOG\_Goal() | Просмотр цели prolog-программы |
| 2 | - | 1 parent( \_, \_ )  2 \_PROLOG\_Goal() | Добавление цели в Call Stack |
| 3 | Parent = "Jerry" | 1 \_PROLOG\_Goal() | Просмотр первого факта родства (Jerry является родителем Kaia). Jerry - родитель. Одно решение цели найдено |

Таблица 4 − Трасса выполнения запроса «Кто является внуком (внучкой)?»

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 1 | Grandchild = \_ | 1 \_PROLOG\_Goal() | Просмотр цели prolog-программы |
| 2 | - | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Добавление цели в Call Stack |
| 3 | X = \_  Y = \_  Z = \_ | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Просмотр первого правила программы (правило определения внуков). Имеем три переменные X, Y, Z, которые нужно определить. |
| 4 | - | 1 parent( \_, \_ )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack правила родства. |
| 5 | X = "Kaia"  Y = "Jerry"  Z = \_ | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Нахождение 1 факта родства. X=Kaia, Y=Jerry |
| 6 | - | 1 parent( \_, "Jerry" )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения родства, где X=Jerry, Y=неизвестно. Проверка утверждения. |
| 7 | - | 1 parent( \_, \_ )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Утверждение неверно. Поиск другого факта родства. |

Продолжение таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 8 | X = "Balto"  Y = "Jerry"  Z = \_ | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Нахождение 2 факта родства. X= Balto, Y=Jerry |
| 9 | - | 1 parent( \_, "Jerry" )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения родства, где X=Jerry, Y=неизвестно. Проверка утверждения. |
| 10 | - | 1 parent( \_, \_ )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Утверждение неверно. Поиск другого факта родства. |
| 11 | X = "Oreo"  Y = "Nina"  Z = \_ | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Нахождение 3 факта родства. X= Oreo, Y= Nina |
| 12 | - | 1 parent( \_, "Nina" )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения родства, где X= Nina, Y=неизвестно. Проверка утверждения. |
| 13 | - | 1 parent( \_, \_ )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Утверждение неверно. Поиск другого факта родства. |

Окончание таблицы 4

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 14 | X = "Balto"  Y = "Nina"  Z = \_ | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Нахождение 4 факта родства. X= Balto, Y= Nina |
| 15 | - | 1 parent( \_, "Nina" )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения родства, где X= Nina, Y=неизвестно. Проверка утверждения. |
| 16 | - | 1 parent( \_, \_ )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Утверждение неверно. Поиск другого факта родства. |
| 17 | X = "Kaia"  Y = "Balto"  Z = \_ | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Нахождение 5 факта родства. X=Kaia, Y= Balto |
| 18 | - | 1 parent( \_, "Balto" )  2 grandchild ( \_ )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения родства, где X= Balto, Y=неизвестно. Проверка утверждения. |
| 19 | X = "Kaia"  Y = "Balto"  Z = "Jerry" | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Нахождение факта 2 родства. Утверждение верно. X=Kaia, Y=Balto, Z=Jerry. |
| 20 | Grandchild = "Kaia" | 1 grandchild ( \_ )  2 \_PROLOG\_Goal() | Kaia - внучка. Одно решение цели найдено |

Таблица 5 − Трасса выполнения запроса «Balto – собака?»

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 1 | - | 1 \_PROLOG\_Goal() | Просмотр цели prolog-программы |
| 2 | X = "Balto"  Y = \_ | 1 dog( "Balto" )  2 \_PROLOG\_Goal() | Добавление цели в Call Stack. Просмотр второго правила программы (правила родства). X= Balto, Y=неизвестно |
| 3 | - | 1 parent( \_, "Balto" )  2 dog( "Balto" )  3 \_PROLOG\_Goal() | Добавление в Call Stack утверждения parent( \_, "Balto" ). Проверка утверждения. |
| 4 | X = "Balto"  Y = "Jerry" | 1 dog( "Balto" )  2 \_PROLOG\_Goal() | Просмотр 2 факта родства. X= Balto, Y= Jerry |
| 5 | X = " Jerry "  Y = \_ | 1 dog( "Jerry" )  2 dog( "Balto" )  3 \_PROLOG\_Goal() | Нахождение 1 факта родства. X=Kaia, Y=Jerry. |
| 6 | - | 1 parent( \_, "Jerry" )  2 dog( "Jerry" )  3 dog( "Balto" )  4 \_PROLOG\_Goal() | Добавление в Call Stack утверждения, что Jerry - собака. Добавление в Call Stack утверждения родства parent(\_, "Jerry" ). Проверка утверждения. |
| 7 | - | 1 dog( "Jerry" )  2 dog( "Balto" )  3 \_PROLOG\_Goal() | Утверждение не верно. Проверка утверждения, что Jerry - собака. |

Окончание таблицы 5

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 8 | X = "Balto"  Y = "Jerry" | 1 dog( "Balto" )  2 \_PROLOG\_Goal() | Утверждение верно. Утверждение родства X= Balto, Y= Jerry верно. |
| 9 | - | 1 dog( "Balto" )  2 \_PROLOG\_Goal() | Утверждение Balto - собака верно, цель найдена. |

Таблица 6 − Трасса выполнения запроса «Кто родитель собаки Balto?»

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 1 | Parent = \_ | 1 \_PROLOG\_Goal() | Просмотр цели prolog-программы. X = Balto, Y =неизвестно. |
| 2 | - | 1 parent( \_, "Balto" )  2 \_PROLOG\_Goal() | Добавление цели в Call Stack |
| 3 | Parent = "Jerry" | 1 \_PROLOG\_Goal() | Просмотр второго факта родства. Y=Jerry, X=Balto. Jerry - родитель. Одно решение цели найдено. |

Таблица 7 − Трасса выполнения запроса «Кому Balto является родителем?»

|  |  |  |  |
| --- | --- | --- | --- |
| № шага | Переменная | Стек вызовов | Пояснение работы |
| 1 | Child = \_ | 1 \_PROLOG\_Goal() | Просмотр цели prolog-программы. Y= Balto, X=неизвестно. |
| 2 | - | 1 parent( "Balto", \_ )  2 \_PROLOG\_Goal() | Добавление цели в Call Stack |
| 3 | Child = "Kaia" | 1 \_PROLOG\_Goal() | Нахождение 5 факта родства. Y= Balto, X=Kaia. Kaia - дочка. Одно решение цели найдено. |

**ПРИЛОЖЕНИЕ Г**

**ПРОГРАММА ОПИСАНИЯ РОДСТВЕННЫХ СВЯЗЕЙ СЕМЬИ**

domains

name=string

predicates

parent(name, name)

man(name)

woman(name)

married(name, name)

grandparent(name, name)

mouther(name, name)

father(name, name)

sister(name, name)

brother(name, name)

cousin(name, name)

niece(name, name)

nephew(name, name)

brotherinlow(name,name)

clauses

grandparent(X,Y) :- parent(Z,Y), parent(X,Z).

mouther(X,Y) :- parent(X,Y), woman(X).

father(X,Y) :- parent(X,Y), man(X).

niece(X,Y) :- sister(Z,Y), parent(Z,X), woman(X).

niece(X,Y) :- brother(Z,Y), parent(Z,X), woman(X).

nephew(X,Y) :- sister(Z,Y), parent(Z,X), man(X).

nephew(X,Y) :- brother(Z,Y), parent(Z,X), man(X).

brotherinlow(X,Y) :- sister(Z,Y), married(X,Z).

cousin(X,Y) :- parent(Z,X), sister(V,Z), parent(V,Y).

cousin(X,Y) :- parent(Z,X), brother(V,Z), parent(V,Y).

sister(X,Y) :- parent(Z,X), parent(Z,Y), woman(Z), woman(X), X <> Y.

brother(X,Y) :- parent(Z,X), parent(Z,Y), woman(Z), man(X), X <> Y.

woman(X) :- married(Y,X).

woman("Валентина").

woman("Вероника").

woman("Анна").

woman("Варвара").

man(X) :- married(X,Y).

man("Александр").

man("Клим").

parent("Валентина", "Борис").

parent("Борис", "Вероника").

parent("Борис", "Светлана").

parent("Евгения", "Вероника").

parent("Евгения", "Светлана").

parent("Светлана", "Анна").

parent("Светлана", "Таня").

parent("Михаил", "Анна").

parent("Михаил", "Таня").

parent("Владислав", "Михаил").

parent("Владислав", "Сергей").

parent("Любовь", "Михаил").

parent("Любовь", "Сергей").

parent("Сергей", "Люба").

parent("Сергей", "Александр").

parent("Татьяна", "Люба").

parent("Татьяна", "Александр").

parent("Анатолий", "Василий").

parent("Люба", "Варвара").

parent("Люба", "Клим").

parent("Олег", "Клим").

married("Борис", "Евгения").

married("Владислав", "Любовь").

married("Михаил", "Светлана").

married("Дмитрий", "Таня").

married("Сергей", "Татьяна").

married("Олег", "Люба").

goal

nl, write(" Список дедушек:"), nl,

grandparent(Дедушка,\_для), man(Дедушка);

nl, write(" Список мам и их детей:"), nl, mouther(Мама,\_для);

nl, write(" Список отцов и их детей:"), nl, father(Папа,\_для);

nl, write(" Список сестер:"), nl, sister(Сестра,\_для);

nl, write(" Список братьев:"), nl, brother(Брат,\_для);

nl, write(" Список двоюродных сестер:"), nl,

cousin(Двоюродная\_сестра,\_для),woman(Двоюродная\_сестра);

nl, write(" Список племянниц:"), nl,niece(Племянница,\_для);

nl, write(" Список племянников:"), nl,nephew(Племянник,\_для);

nl, write(" Список зятей:"),nl,brotherinlow(Зять,\_для).

%write("Для Варвары "), grandparent(Дедушка,"Варвара"), man(Дедушка).

%write("Для Анны "), father(Папа,"Анна").

%write("Для Анны "), mouther(Мама,"Анна").

%write("Для Анны "), cousin(Двоюродная\_сестра,"Анна"),woman(Двоюродная\_сестра).

%write("Для Анны "), brotherinlow(Зять,"Анна").