Министерство образования и науки РФ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Уральский государственный педагогический университет»
Институт математики, информатики и информационных технологий
Кафедра информатикиинформационных технологий и методики обучения
информатике

АНАЛИЗ ФОРМАЛЬНЫХ ПОНЯТИЙ КАК СРЕДСТВО СОЗДАНИЯОБУЧАЮЩЕЙИНТЕЛЛ ЕКТУАЛЬНОЙ СИСТЕМЫ

Выпускная квалификационная работа
бакалавра по направлению подготовки
02.03.02 – Фундаментальная информатика и информационные технологии

Исполнитель:
студент группы Б-41
Института математики, информатики и ИТ
Давыдова Е. А.

Руководитель:
д.п.н., зав.кафедройИИТиМОИ
ЛапенокМ. В.

Работа допущена к защите
«____»_____ 2017 г.
Зав. кафедрой _____

Екатеринбург – 2017

Реферат

Давыдова Е.А. АНАЛИЗ ФОРМАЛЬНЫХ ПОНЯТИЙ КАК СРЕДСТВО СОЗДАНИЯ ОБУЧАЮЩЕЙ ИНТЕЛЛЕКТУАЛЬНОЙ СИСТЕМЫ, выпускная квалификационная работа: 55 стр., рис.14, табл.1, библ. 30 назв., приложений 2.

Ключевые слова: АНАЛИЗ ПОНЯТИЙ, ИНТЕЛЛЕКТУАЛЬНЫЕ СИСТЕМЫ, СЕМАНТИЧЕСКИЕ СЕТИ, ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ, ПРОЛОГ.

Объект разработки: процесс создания интеллектуальных учебных систем.

Цель работы: разработать программный комплекс, моделирующий интеллектуальную деятельность по анализу текста учебных задач посредством выявления в тексте формальных понятий, и автоматизирующий процесс создания Пролог-программ для решения учебных задач.

Работа направлена на разработку с использованием языков *ObjectPascal* и *VisualProlog* программного комплекса, автоматизирующего процесс создания учебных программ, ориентированных на решение логических задач. В работе описана методика анализа формальных понятий и их группировки в зависимости от параметров объектов предметной области, применение которой позволило обосновать алгоритмы генерации правил представления предметной области в программных комплексах.

Оглавление

Введение

На мировом рынке IT-технологий наблюдается серьезная нехватка молодых специалистов, которая исчисляется сотнями тысяч. Это связанно с тем, что в наше время программирование является невероятно важным навыком, а работа разработчика программ, в том числе и программ на языке логического программирования, одна из наиболее востребованных в мире. В связи с этим выявлена необходимость подготовки квалифицированных специалистов высших учебных заведений для дальнейшей профессиональной деятельности.

В процессе подготовкиспециалистов в настоящее время целесообразно использовать методические приёмы, способствующие повышению интереса обучающихся к изучению логического программирования, исредства обучения, реализующие автоматизацию создания Пролог-программ. В качестве средства моделирования предметных областей при решении учебных задач логического программированияможно использовать анализ формальных понятий, который позволяет в тексте учебной задачи выявить ключевые слова и использовать их при генерации шаблонов Пролог-программ. Таким образом, можно автоматизировать следующие виды интеллектуальной деятельности человека:

- деятельность по анализу текста (формулировки) задачи с целью выявления ключевых слов и определения типа учебной задачи;
- деятельность по анализу ключевых слов в формулировке задачи с целью определения структуры Пролог-программы;
- деятельность по генерации шаблона Пролог-программы на основе выявленной структуры;
- деятельность по написанию программ на языке Пролог в процессе решения учебных задач.

Объект исследования: процесс создания интеллектуальных учебных систем.

Предмет исследования: применение анализа формальных понятий дляавтоматизации создания Пролог-программ, направленных на решение учебных задач.

Цель: разработать программный комплекс, моделирующий интеллектуальную деятельность по анализу текста учебных задач посредством выявления в тексте формальных понятий,и автоматизирующий процесс создания Пролог-программ для решения учебных задач.

Для достижения цели были поставлены следующие задачи:

1. Проанализировать научно-методическую литературу в области методов и средств моделирования предметных областей.
2. Разработать типизацию учебных задач для языка логического программирования Пролог,исследовать возможность применения анализа формальных понятий для их решения и составить комплекс учебных задач.
3. Разработать методику анализа формальных понятий и их группировки в зависимости от параметров объектов предметной области.
4. Разработать методику генерации правил представления предметной области основанную на объявлении предикатов с одним или несколькими параметрами.
5. Создать с использованием языков *ObjectPascal* и *VisualProlog* программный комплекс, автоматизирующий процесс разработки учебных программ, ориентированных на решение логических задач.

Глава 1. Системы автоматизации интеллектуальной деятельности

1.1 Интеллектуальные системы и технология их создания

Интеллектуальная система представляет собой компьютерную модель интеллектуальной деятельности человека целенаправленного поиска, анализа и обобщения текущей информации об окружающей среде для получения новых о ней знаний и на основе этих знаний решения ряда жизненно важных задач [22,24].

Исследователи, решающие задачи в данном направлении, намерены достигать такого понимания механизмов интеллекта, который представит собой компьютерные программы с человеческим интеллектом или интеллектом более высокого уровня. Общий подход заключается в разработке методов по решению задач, в которых не существуют формальные алгоритмы: понимание естественного языка, доказательство теорем, обучение, распознавание образов и другое. Теоретическое исследование фокусируется на изучении интеллектуальных процессов и создании соответствующих математических моделей. Экспериментальная работа выполняется путем создания машин и компиляции компьютерных программ, которые решают частные интеллектуальные задачи. Подробные исследования в области искусственного интеллекта начали возникать после появления цифрового компьютера. В 1950 году была опубликована А.Тьюрингом первая научная статья по искусственному интеллекту. Развитие интеллектуальных систем на современном этапе проходит в трех направлениях исследований [18].

В рамках первого направления объектом исследования стали структура и механизм работы человеческого мозга для дальнейшего раскрытия тайн мышления. Важным этапом в исследовании этого направления считается построение модели интеллектуальной деятельности человека на основании психофизиологических данных.

Во втором направлении особенностью объекта исследования рассматривается искусственная интеллектуальная система. А именно, речь идет о моделировании интеллектуальной человеческой деятельности при помощи компьютеров. Целью работы данного направления является создание программного обеспечения, которое позволит решать некоторые типы интеллектуальных задач не хуже человека.

А третье из направлений ориентируется на создание интерактивных, интеллектуальных систем. Не маловажной проблемой в этих исследованиях является организация семантически совершенного диалога человека с системой [12,23].

В настоящее время при построении систем искусственного интеллекта и выборе метода представления знаний используются различные подходы или их комбинации, среди которых[21]:

- логический подход;
- структурный подход;
- эволюционный подход;
- имитационный подход.

Возникновение *логического подхода* напрямую связано с развитием логического мышления у человека[25]. Основам данного подхода положили начало логика Аристотеля, исчисления высказываний и Булева алгебра. Затем были введены новые понятия формальных систем аксиом, и на базисе этого развивалось классическое исчисление предикатов. Исходные данные представляются в базах знаний в виде правил логического вывода отношений, в виде аксиом и фактов, а также в базах данных. Все интеллектуальные системы, построенные на логике, реализуют машины доказательств теорем, которые имеют систему вывода для доказательства цели как теоремы и блок для генерации этой самой цели. Используя трассировку применяемых правил для реализации сгенерированной цели можно получить необходимую цепочку действий, при условии, что цель удалось доказать.

Структурный подход. Его название связано с попытками построения искусственного интеллекта посредством моделирования на вычислительной машине структуры мозга человека, содержащей модели нейронов мозга и нейронных сетей. Первая простейшая реализация – это персептрон Ф. Розенблатта для распознавания зрительного образа. Затем это направление стало развиваться в теорию распознавания образов, так называемые «искусственные нейронные сети».

А также достаточно широко используется *эволюционный подход* при построении систем искусственного интеллекта. В данном подходе всё внимание концентрируется на построении начальной модели и правилах, по которым она может поменяться (эволюционировать), исходя из этого, модель может быть составлена благодаря самым различным методам, включая нейронные сети, наборы логических правил и любые другие модели. Вместе с правилами в программе определяются ещё и критерии оценки качества каждого варианта.Запустив программу, с начальным вариантом модели, начинается итеративное построение и оценка вариантов. В результате оценки и проверки качества моделей, полученных после каждой итерации, выбираются наиболее успешные из них [28]. На основе этих вариантов и правил создаются новые модели, среди которых снова выбирают самые удачные и так далее.

Имитационный подход в свою очередь основан на введенном У.Р. Эшби классическом базовом понятии кибернетики «черного ящика». Модель данного объекта исследования заключается в его поведении, реакции на воздействия, поступающие извне на его входы, характеризует связи между реакциями и вызвавшими их воздействиями и внешне имитирует способность человека копировать поведение других, не понимая, как это происходит и почему. Единственным минусом имитационного подхода (также как эволюционного) стала низкая информационная способность большинства созданных моделей (в смысле понимания структуры и параметров модели «внутреннего» устройства черного ящика)[7].

По мнению современных исследователей (Афонина В.А., Макушкина В.А.) рассмотрены и описаны задачи, к которым применяются выделенные подходы. Такв принятии решений на верхнем уровне управления чаще всего используется логический подход, а задачами систем управления нижних уровней является обеспечение взаимодействия интеллектуальной системы с внешней средой, то есть формирование реакций интеллектуальной системы в зависимости от решения, получение и первичная обработка информации, принятого на верхнем уровне. Таким образом, в системах нижнего уровня, в которых осуществляется обработка первичной информации, в основном используются структурный, эволюционный или имитационный подходы и соответствующие им решения, «подсмотренные» человеком в живой природе. Что позволит на данном этапе развития организовать более эффективное распознавание звуковых, зрительных и тактильных первичных информационных образов, а также решать задачи идентификации, моделирования развития ситуаций в реальном и ускоренном времени и других задач [2].

Таким образом, в данной работе целесообразнее всего использовать логический подход построения интеллектуальной обучающей системы, так как исходные данные представляются в базах знаний в виде правил логического вывода отношений, в виде аксиом и фактов, а также в базах данных.

1.2 Семантические сети как средство моделирования предметной области в интеллектуальных системах

Одной из наиболее важных проблем, с которой приходится сталкиваться при построении систем, основанных на знаниях, является представление знаний. Это связано с тем, что представление знаний в конечном итоге во многом определяет характеристики системы. База знаний создается для ввода знаний и обучения системы (часто называемого приобретением знаний) и

последующей обработки знаний для решения задач. Способ представления знаний часто называют *моделью представления знаний*[14,29].

При выборе надлежащего способа представления знаний можно избежать усложнения системы и разрешить множество вопросов ее проектирования и разработки. Однако выбор оптимального способа представления знаний для конкретной задачи сильно зависит от характера и сложности этой задачи. Более того, определение модели представлений знаний накладывает ограничения на выбор соответствующего механизма логического вывода. То есть, при проектировании интеллектуальной системы необходимо выбирать наиболее подходящую модель представления знаний (и в соответствии с ней выбирать используемый механизм логического вывода)[17].

При проектировании модели представления знаний следует учитывать такие факторы, как однородность представления и простота понимания. Однородное представление приводит к упрощению механизма управления логическим выводом и упрощению управления знаниями. В то же время представление знаний должно быть понятным специалистам, экспертам и пользователям системы. В противном случае затрудняются приобретение знаний и их оценка. Однако выполнить эти требования в равной степени как для простых, так и для сложных задач довольно трудно. Обычно для несложных задач останавливаются на некотором среднем (компромиссном) представлении, но для решения сложных и больших задач необходимы структурирование и модульное представление. Поэтому необходимо ознакомиться с типовыми моделями представления знаний и рассмотреть особенности каждой из них. Типовые модели представления знаний:

- логическая модель;
- продукционная модель;
- модель, основанная на использовании фреймов;
- модель семантической сети.

Логическая модель представления знаний часто называется представлением знаний с использованием логики предикатов. В ее основе лежит язык математической логики, позволяющий формально описывать понятия предметной области и связи между ними [23].

В отличие от естественного языка, который очень сложен, язык логики предикатов использует только такие конструкции естественного языка, которые легко формализуются. То есть логика предикатов – это языковая система, которая оперирует с предложениями в пределах жесткого набора синтаксических правил этого языка.

Продукционные модели – наиболее простой способ, представления знаний. Он основан на представлении знаний в форме правил, структурированных в соответствии с образцом «ЕСЛИ - ТО». Часть правила «ЕСЛИ» называется посылкой, а «ТО» – выводом или действием. Правило в общем виде записывается так:

ЕСЛИ A1, A2, ...,An, ТО B.

Такая запись означает, что «если все условия от A1 до An являются истинными, то B также истинно» или «когда все условия от A1 до An выполняются, то следует выполнить действие B».

Программные средства, оперирующие со знаниями, представленными правилами, получили название продукционных систем (или систем продукции) и впервые были предложены Постом в 1941 году [6].

Фреймовая система имеет все свойства, присущие языку представления знаний, и одновременно являет собой новый способ обработки информации. Слово «фрейм» в переводе с английского языка означает «рамка».

Фрейм является единицей представления знаний об объекте, которую можно описать некоторой совокупностью понятий и сущностей. Он имеет определенную внутреннюю структуру, состоящую из множества элементов, называемых слотами. А каждый слот, в свою очередь, представляется

определенной структурой данных, процедурой, или может быть связан с другим фреймом [6].

Стандартного определения семантической сети не существует, но обычно под ней подразумевают следующее:

Семантическая сеть– это система знаний, имеющая определенный смысл в виде целостного образа сети, узлы которой соответствуют понятиям и объектам, а дуги – отношениям между объектами [23].

При выборе оптимального способа представления знаний мы пришли к выводу, что семантические сети больше всего подходят для достижения цели в данной работе, так как в основе этого способа представления знаний лежит идея о том, что любые знания можно представить в виде совокупности понятий (объектов) и отношений (связей).

Основным преимуществом этой модели является наглядность и соответствие современным представлениям об организации долговременной памяти человека. Недостаток – сложность поиска вывода, а также сложность корректировки, то есть удаления и заполнения сети новыми знаниями.

На протяжении более пятидесяти лет велась разработка, и совершенствовались различные инструментальные средства для повышения эффективности в создании, отладке и совершенствовании интеллектуальных систем. В свою очередь, создание и развитие самих инструментальных средств разработки искусственного интеллекта происходило в двух направлениях.

Одно направление составило универсальные инструментальные средства – в основном языки программирования для искусственного интеллекта и некоторые универсальные языки программирования. Эти средства чаще всего применяются при создании различных системискусственного интеллекта, которые основаны на знаниях.

Второе направление – специализированные программные средства, конфигураторы баз данных, баз знаний, специальные языки и различные редакторы для поддержки моделей представления знаний вискусственном

интеллекте. Данные средства позволяют более эффективно создавать экспертные системы, строить системы интеллектного управления на основе использования нечеткой логики, искусственных нейронных сетей, генетических алгоритмов и т.п.

В процессе развития искусственного интеллекта было предпринято много различных попыток при создании разнообразных инструментальных средств, построенных на основе различных идей, методов и подходов. Дальнейшее развитие и применение получили не все методы и подходы, только некоторые доказали свою полезность и эффективность и до сих пор активно используются при создании систем искусственного интеллекта [19].

Рассмотрим первое направление – применение языков программирования для создания искусственного интеллекта.

Разработчики языков делят алгоритмические языки на две группы. Первая группа, образующая языки, названа языками операторного, или процедурного типа. Элементарными единицами программы здесь являютсяоператоры, так называемые, приказы, выполнение которых сводится к определенному изменению конкретной части памяти машины. Представителем данной группы является язык машины Поста. А также сюдавключаются машинные языки конкретных ЭВМ и массовые языки программирования типа Фортран, Алгол, ПЛ/1, С++.

Языки второй группы - это языки сентенциального, или декларативного типа (sentence– предложение, высказывание). Программа на таком языке представляет собой набор предложений (соотношений, правил, формул), которые машина, понимающая данный язык, умеет применять к обрабатываемой информации. Простым примером сентенциального языка, созданного с теоретическими целями, является язык нормальных алгоритмов Маркова.

Можно выделить прообразы указанных типов алгоритмических языков в естественных языках. Для операторных языков повелительное наклонение

(приказание, императив), а для сентенциальных – изъявительное наклонение (повествование, описание). Обращаясь к естественному языку, очевидно, что изъявительное наклонение является более распространенным и образует основу языка, в то время как повелительное наклонение определяется некоторой специальной модификацией. Следовательно, можно сделать вывод о том, что «относительный вес изъявительного наклонения является мерой развитости языка» [20].

Основными языками программирования в области искусственного интеллекта на данном этапе считаются:ЛИСП и ПРОЛОГ [1,4,5],которые являются узкоспециализированными языками программирования, специально разработанными для искусственного интеллекта.

Применение LISPa и ПРОЛОГа позволяет разработчику систем искусственного интеллекта сосредоточиться на логике решения задачи.

Программа-оболочка, или матрица, называемая иногда «Пустой экспертной системой», – это программа, содержащая уже механизм вывода, средства ввода и редактирования фактов, средства объяснения и не содержащая конкретных фактов конкретной предметной области. Разработчики ЭС могут наполнить Матрицу собственными фактами, правилами, отредактировать правила, и она начнет делать выводы по интересующей Вас проблеме[9,13].

1.3 Техническое задание на разработку программного комплекса

Составлено на основе ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы»

1. Общие сведения.

1.1. Название организации-заказчика: Уральский Государственный Педагогический Университет (УРГПУ).

1.2. Название продукта разработки (проектирования): программный комплекс «Тренажёр-Практикум для освоения языка программирования

Пролог», обеспечивающий написание программ на языке логического программирования Пролог.

1.3. Назначение продукта: программный комплекс «Тренажёр-Практикум для освоения языка программирования Пролог»обеспечивает условия (посредством предоставления шаблонов программ) для написания программ на языке логического программирования Пролог для решения (реализации) учебных задач различных типов (в том числе типов: логические задачи, арифметические вычисления, базы знаний и др.). При этомвыполняется автоматизированный анализ текстовой формулировки задачи, выделяются ключевые слова, на основании которых формируется шаблон определенного типа. В шаблоне описание предметной области представлено в форме набора фактов (на основе предикатов определённой арности), а также в форме совокупности правил, построенных в соответствии с разработанной методикой.

2. Характеристика области применения продукта.

2.1. Процессы и структуры, в которых предполагается использование продукта разработки: продукт разработки предполагается использовать в учебном процессе в качестве средства, способствующего освоению дисциплин «Интеллектуальные системы», «Основы искусственного интеллекта», «Интеллектуальные системы и технологии», «Программная инженерия», «Программирование» и «Моделирование интеллектуальных систем».

2.2. Характеристика персонала: для использования данного продукта в учебном процессе вуза не требуются вспомогательный персонал и особые навыки обращения с компьютером, а только базовые знания.Перед началом применения продукта (однократно) лаборант или преподаватель должен поместить файл приложения на сетевом диске и/или локальном диске компьютера, чтобы студенты имели свободный доступ к ресурсу.

3. Требования к продукту разработки.

3.1. В целом к готовому продукту предъявляются следующие требования: круглосуточная возможность работы с ресурсом, не требующая доступа к сети Интернет. Продукт должен представлять собой программный комплекс с пользовательским интерфейсом, разработанный в среде программирования *Delphi 7* и в среде логического языка программирования Пролог.

3.2. Аппаратные требования. Персональный компьютер, со следующими характеристиками:

- процессор с тактовой частотой не менее 300 МГц;
- оперативная память: 1Гб;
- свободное место на жестком диске около 5 Мбайт;
- монитор с разрешением не менее 800 на 600 точек;
- устройства ввода: компьютерная мышь, клавиатура.

3.3. Указание системного программного обеспечения: *VisualProlog* поддерживается различными ОС, в том числе *MS-DOS PharLap-Extended DOS*, всеми версиями *Windows*, 16- и 32-битовыми целевыми платформами *OS/2*, а также некоторыми другими системами, требующими графического пользовательского интерфейса[5].

3.4. Указание программного обеспечения, используемого для реализации: текстовый процессор *MicrosoftOfficeWord*, текстовый редактор «Блокнот» или любой другой текстовый редактор позволяющий работать с файлами формата **.txt*, для создания и размещения продукта необходима интерактивная среда визуальной разработки (*VDE — VisualDevelopEnvironment) VisualProlog 5.2* и среда программирования *Delphi7*.

3.5. Форматы входных и выходных данных: входные данные – текст задачи, которую необходимо реализовать, выходные данные – сгенерированный шаблон, подходящий для решения определённой задачи.

3.6. Источники данных и порядок их ввода в программу, порядок вывода, хранения в программном комплексе:

- после запуска приложения, на форме приветствия появляется раскрывающееся сбоку меню, которое позволяет начать работу с программой;
- в первом блоке выводятся примеры задач с демонстрацией структуры готовой программы, которые позволяют изучить типовые учебные задания для дальнейшего обучения логическому программированию;
- во втором блоке составлен комплекс задач для самостоятельной работы, которые анализируются программой для генерации шаблона, также формулировку задания можно ввести вручную, после анализа на экран выводится тип задачи и генерируется подходящий шаблон;
- затем открывается шаблон, после запуска Пролога и начинается непосредственное решение задачи.

3.7. Порядок взаимодействия с другими системами, возможности обмена информацией: взаимодействие с другими системами происходит при нажатии кнопки «Запуск Prolog» в среде программирования *Delphi7*, после чего запускается среда разработки *VisualProlog 5.2* для дальнейшей работы. Также при вызове справочной информации по работе с программой.

3.8. Меры защиты информации: программа предоставляется в виде отдельного *.exe* файла без исходного кода. Поэтому доступом к редактированию содержимого приложения владеет только разработчик.

4. Требования к пользовательскому интерфейсу.

4.1. Общая характеристика пользовательского интерфейса.

В требованиях к программной реализации выделяются следующие пункты:

- простота в использовании, интуитивно понятный и комфортный для пользователя интерфейс;
- управляющие элементы строго выровнены;
- абсолютная симметричность затрудняет видение и восприятие информации с экрана;

- все элементы одной категории одинакового цвета и размера, для однозначного определения их принадлежности;
- все компоненты приложения должны быть сдержанных цветов, в одной цветовой гамме и дополнять друг друга, так как слишком агрессивные, могут отвлекать внимание пользователя, создавать трудности в работе.

4.2. Размещение информации на экране, дизайн экрана.

Общие принципы расположения информации на экране должны обеспечить для пользователя:

- возможность просмотра экрана в логической последовательности;
- простоту выбора нужной информации;
- возможность идентификации связанных групп информации;
- различимость исключительных ситуаций (сообщений об ошибках или предупреждений);
- возможность определить, какое действие со стороны пользователя требуется (и требуется ли вообще) для продолжения выполнения задания.

4.3. Особенности ввода информации пользователем, представление выходных данных.

Способы ввода данных, реализованные в программном комплексе, соответствуют требованиям к управлению данными. Атрибуты, используемые применительно к этому аспекту оценки ресурсов, должны включать в себя: удобный формат ввода данных, проверку достоверности данных и обеспечение совместимости с другими структурами данных.

Способы вывода данных, реализованные в программном комплексе, соответствуют требованиям к доступности терминологии. Они характеризуют непосредственные информационные нужды пользователей в следующих аспектах: читабельность, доступ и пользовательский отбор данных.

5. Требования к документированию.

5.1. Перечень сопроводительной документации.

- Техническое задание.

- Руководство пользователя.

5.2. Требования к содержанию отдельных документов.

Техническое задание – это исходный документ для проектирования информационной системы. Оно должно содержать технические требования, предъявляемые к продукту и исходные данные для разработки; в нем указывается информация о назначении объекта, области его применения, стадии разработки документации, её состав, сроки выполнения.

Руководство пользователя – это документ, содержащий инструкции, описание действий для помощи пользователю в использовании системы и для ее полноценной и корректной работы.

Обычно руководство пользователя содержит такие пункты как:

1) Аннотация, в которой приводится краткое изложение содержимого документа и его назначение.

2) Введение, содержащее ссылки на связанные документы и информацию о способе эффективного использования данного руководства.

3) Страницу содержания.

4) Главы, описывающие, как использовать, наиболее важные функции системы.

5) Глава, описывающая возможные проблемы и пути их решения.

6) Указание мест поиска дополнительной информации по предмету, контактная информация.

7) Предметный указатель.

6. Порядок сдачи-приемки продукта.

6.1. Промежуточный контроль – март 2017г., объем – основной функционал, контроль – руководитель.

6.2. Дата отчета руководителю – начало мая 2017г.

Глава 2. Обучающая интеллектуальная системадля освоения языка программирования Пролог

2.1 Обоснование выбора программных средств для разработкипрограммного комплекса

Язык логического программирования ПРОЛОГ[1].

ПРОЛОГ (PROgramminginLOGic) разработан в 1974 г. в университете Марселя (Франция) А. Колмаэро на основе фундаментальных работ А. Робинсона и Р. Ковальского. Рассмотрим основные сведения о языке.

Все переменные в Пролог-программе пишут с большой буквы: X, Misha. Если значение переменной нас не интересует, используют пустую переменную «_». Свободной называется переменная, значение которой еще не определено.

Основные раздела объявления констант:

- [GLOBAL] domains – секция объявления нестандартных и/или составных типов данных. Может отсутствовать;
- [GLOBAL] database(имя_ВБД) –необязательная секция объявления предикатов для работы с внутренней базой данных;
- [GLOBAL] predicates – секция объявления предикатов;
- Clauses – секция объявления правил и фактов;
- Goal – секция объявления внутренней цели. Может отсутствовать.

Типы данных:

- строка, занесенная во внутреннюю таблицу символов системы(symbol);
- последовательность символов длиной до 64 Кбайт(string);
- 1-байтовые символы(char);
- 2-байтовые целые числа со знаком(integer);
- 8-байтовые числа с плавающей точкой(real);
- ссылочные числа базы данных(ref);
- регистры микропроцессора AX, BX, CX, DX, SI, DI,DS и ES(reg);

- файл(file).

Рассматриваются четыре типа операций, а именно: арифметические (+, -, *, /, mod, div), реляционные (>, <, =, >=, ,<=, <>, ><), математическиефункции (sin, cos, tan, arctan, ln, log, exp, sqrt, round, trunk, abs) и логические (And(«,»), not, or, !(отсечение)).

Ввод-вывод осуществляется операциями write (вывод на экран) и read (чтение с консоли).

Рассмотрим пример конкретной версиипрограммы (TurboProlog):

```
domains
        person, activity = symbol
predicates
        likes(person,activity)
clauses
        likes(ellen,tennis)
        likes(john,football)
        likes(tom,baseball)
        likes(bill,X)
        likes(tom,X)
goal
        likes(X,У)
```

Результатработы (произошлаконкретизациязначенийпеременныхпоимениотношения, с помощьюпредикатalikes (нравится)):

```
X = ellenY = tennis
X = johnY = football
X = tomY = baseball
X = billY = baseball
```

Язык функционального программирования ЛИСП [30].

Язык создан в 1961 г. группой Стэнфордского профессора Джона Маккартни в США. Сокращение ЛИСП [LISP (Listprocessing)] трактуетсякак «язык обработки списков». В 70-80 гг. широко использовался для решения задач основанных на древовидных структурах, например, задач лабиринтного поиска и «генетического программирования». Существует не малое количество версий, наиболее известная из них, как мы видим, COMMON LISP. Эта версия была поддержана AILab M.I.T при создании LISP-машины в качестве языка системного программирования. К началу 90-х гг. в Европе фактически вышел из эксплуатации, применяется в США.

Lisp-программа не имеет конкретной структуры. Представляет собой последовательность s-выражений (т. е. символьных, в заданной грамматике), поступающих последовательно на вход интерпретатора Lisp.В таблице 1 представлены типы данных.

Таблица 1. Типы данных

Целое число	Например 4	
Вещественное число	Определяется по наличию точки – 4.5	
Символьный атом	Аналог понятия переменной. Значение переменной по умолчанию равно самой переменной	Так называемые (Symbolic) - выражения
Список	Обозначается (), например (A, B, C)	
Встроенная функция	Список приведен ниже	
Примитив	Толко для AutoLisp	

Пример программы (μ -LISP).

Вычислениефакториала:

```
(defun factorial (n)
(cond ((= n 1) 1)
(t (* n (factorial (- n 1))))))
```

Ряд специалистов считает, что Пролог более удобен в качестве учебного языка логического программирования, а Лисп является языком функционального программирования, который относится к профессиональному средству. Но они считаются равнозначными с точки зрения быстродействия и вычислительной мощности. Программа на Лисп значительно короче, но Пролог – программа понятнее. Последнее обстоятельство важно для программистов, так как размер программы сильно ограничен не памятью машины, а способностью программиста понимать, «что же он делает». Существуют несколько различных версий языка: *TurboProlog*, *VisualProlog* и др. Одни версии имеют ограничения и больше подходят для начального освоения, другие обеспечивают полную профессиональную поддержку работы программиста. В учебном процессе были изучены основы Пролога, это упрощает разработку программного комплекса в данной среде.

На основании выше перечисленных преимуществ, языком для разработки программного комплекса – шаблона, былвыбран язык программирования *VisualProlog5.2*. с возможностью создания визуальной части программы (пользовательского интерфейса).

Язык объектно-ориентированного программирования *ObjectPascal* [10].

*ObjectPascal*являетсяпоследнейверсиейсемействаязыков *Pascal*, реализующейпринципыобъектно-ориентированногопрограммирования.Этот языкявляетсяосновойсистемывизуальногопрограммирования*Delphi*. Наиболее существеннымотличием от традиционногоязыка Pascalявляетсяналичиедостаточносложныхструктур данных (классы) и возможностьсредствами Pascalобращатьсякфункциям*WindowsAPI* длясозданияполноценныхWindows-приложений[26].

ObjectPascal позволяетиспользоватьмножество самых разнообразныхтипови структур данных. Все типы данных можно разбить на две группы: простые (базовые) и структурированные (пользовательские) типы [3],

которые создаются на основе базовых и объединяют несколько переменных разных типов в одной структуреданных.

Любые переменные, используемые в программе, должны быть описаны в разделах описаний программы, соответствующих процедур, функций или библиотечных модулей. При описании указывается имя переменной и ее тип. Тип данных определяет:

1) множество допустимых значений, которые может принимать переменная;

2) набор допустимых операций над этой переменной;

3) формат внутреннего представления данных в оперативной памяти, в частности размер памяти, отводимый под хранение переменной [11]. Объявление переменных простых типов производится непосредственно в разделе описания переменных с использованием стандартных идентификаторов. К простым переменным часто относят и переменныестрокового типа(строкасимволов).
Сточкизренияструктурыданныхстроковыетипыявляютсямассивамисимволов, т.е. структурированнымитипами. Однако, учитываяширокоеиспользованиестроквпрограммировании, ихотнесликбазовымтипам, введядлянихстандартныеидентификаторы[10].

Для анализа текста (формулировки) задачи с целью выявления ключевых слов и определения типа учебной задачи было решено разработать приложение на языке ООП. Среди изученных языков программирования предпочтение было отдано *Delphi*, поскольку его компиляторпозволяет внести изменения и за короткое время увидеть полученный результат с малыми экономическими затратами.

Delphi – среда программирования, в которой сочетаются простота и удобство с мощью и гибкостью. Она обеспечивает визуальное проектирование пользовательского интерфейса, что является главным плюсом.

Язык *Delphi*простой и логичный. Все необходимые конструкции языка удобно выделяются в тексте программы, что способствует хорошему восприятию при написании и редактировании программного кода. Также, в среде разработки доступ предоставляется только к конкретным участкам кода, с которыми необходимо работать в данный момент.

Язык программирования *Delphi* полностью удовлетворяет требованиям для разработки программы для автоматизации анализа формальных понятий с выделением ключевых слов и генерации шаблонов типовых Пролог-программ.

2.2 Описание методики анализа формальных понятий и генерации правил представления предметной области в программных комплексах

Анализ формальных понятий является методом анализа данных, который включает анализ следующих множеств: объектов, дескрипторов (характеристик объектов) и отношений (признаков объектов, обусловленных присущими объекту характеристиками) [16]. Для повышения эффективности создания учебных программных комплексов, ориентированных на решение логических задач, целесообразно разработать методику анализа формальных понятий и их группировки в зависимости от параметров объектов предметной области, применение которой позволит обосновать алгоритмы генерации правил представления предметной области в программных комплексах.

Методика анализа формальных понятий и их группировки в зависимости от параметров объектов предметной области основывается на выявлении и обосновании инвариантного набора дескрипторов (характеристик), подходящих для представления различных предметных областей [27]. Например, таких характеристик, как «условие», «ограничение», «соответствие», «выражение» и т.п. На основании анализа характеристик типовых учебных задач и предметных областей были определены четыре набора дескрипторов, соответствующих основным типам учебных задач: задач на выполнение арифметических вычислений, логических задач, задач на анализ значений с последующей

выборкой или упорядочением, задач на создание баз знаний по предметной области. В исследовании были приняты следующие соглашения. Учебная задача была отнесена к первому типу задач – типу «на выполнение арифметических вычислений», если в процессе анализа текста этой учебной задачи в формулировке были выявлены следующие дескрипторы: «вычислить» (или синонимы «подсчитать», «определить», «найти значение», «значение выражения», «является ли ...») и «формула» (или синонимы «аналитическое выражение», «имя переменной =», «имя функции (параметры)=»). Если в формулировке задачи были выявлены дескрипторы «установить соответствие» (или синонимы «распределить по местам», «найти соответствующие пары элементов») и «ограничения» (или синонимы «если <условие>», «факты», «противопоказания», «самоограничения»), то такая задача была отнесена ко второму типу – типу «логические задачи». Если цель задачи состоит в нахождении локального экстремума (крайних элементов ряда), что содержательно выражается дескрипторами «самый …», «найти наименьший элемент», «установить порядок» и «сравнение» (представленное в форме альтернатив «максимум/минимум», «выше/ниже», «больше/меньше», «старше/моложе», «левее/правее»), то такая задача была отнесена к третьему типу – типу «задачи на анализ значений с последующей выборкой или упорядочением». Учебная задача была отнесена к четвёртому типу – типу «задачи на создание баз знаний по предметной области», если в процессе анализа текста в формулировке задачи были выявлены дескрипторы «создать» (или синонимы «реализовать», «сформировать», «разработать») и «база знаний» (или синонимы «справочник», «проект», «база данных», «информация»).

Методика генерации правил представления предметной области основана на объявлении предикатов с одним или несколькими параметрами, которые соотнесены с особенностями формулировок типовых учебных задач. В задачах первого типа «на выполнение арифметических вычислений» целевая

характеристика (т.е. числовая переменная, значение которой определяет условия истинности целевого предиката) представляет собой совокупность предикатов *{write() и read()}*, которые достигают значения «истинность» привычислении формулы и выводе решения на экран. Для решения «логических задач» необходимо установить соответствие между объектами, т.е. определить в программе несколько групп одиночных переменных (или несколько массивов переменных), рассматривая вышеназванные переменные как элементы конечных множеств, между которыми устанавливаются взаимно-однозначные соответствия. При этом количество элементов в соответствующих множествах должно быть одинаковым [4]. В «задачах на анализ значений с последующей выборкой или упорядочением» целевая характеристика (т.е. элементы, последовательность которых определяет условия истинности целевого предиката) представляет собой предикат *ряд()*, который достигает значения «истинность» при нахождении искомого элемента в полученной последовательности и выводе результата на экран. В «задачах на создание баз знаний по предметной области» целевая характеристика (т.е. база фактов, включающая в себя составные объекты) представляет собой совокупность утверждений целевого предиката, который достигает значения «истина» при осуществлении выборки, поиска по критериям или вывода всей базы на экран [8].

2.3 Реализация программного комплекса «Тренажёр-Практикум для освоения языка программирования Пролог»

В процессе исследования было разработано приложение на языке Пролог, реализующее графический интерфейс[15] и функционал генерации шаблона типовых Пролог-программ, предназначенное для освоения логического программирования, которое в свою очередь необходимо для решения логических задач. Активизация объектов (кнопок), размещенных на главной странице разработанного приложения (рис.1) инициирует выполнение примеров Пролог-программ, соответствующих типовым задачам (рис.2, рис.3).

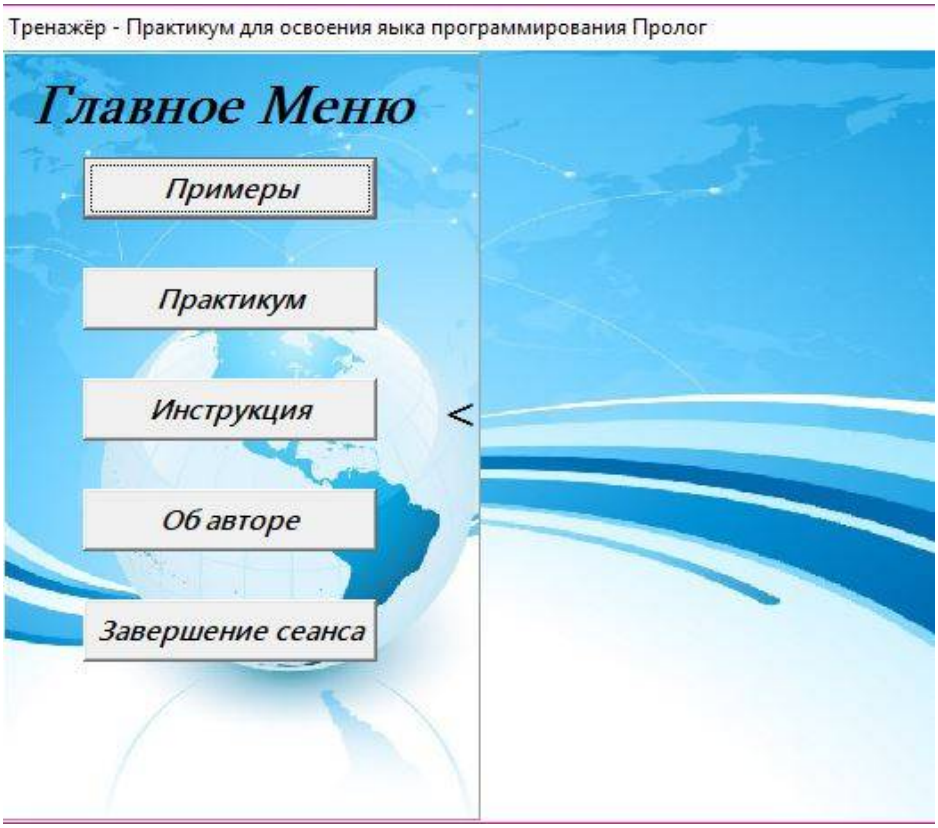

Рис. 1. Интерфейс созданного приложения

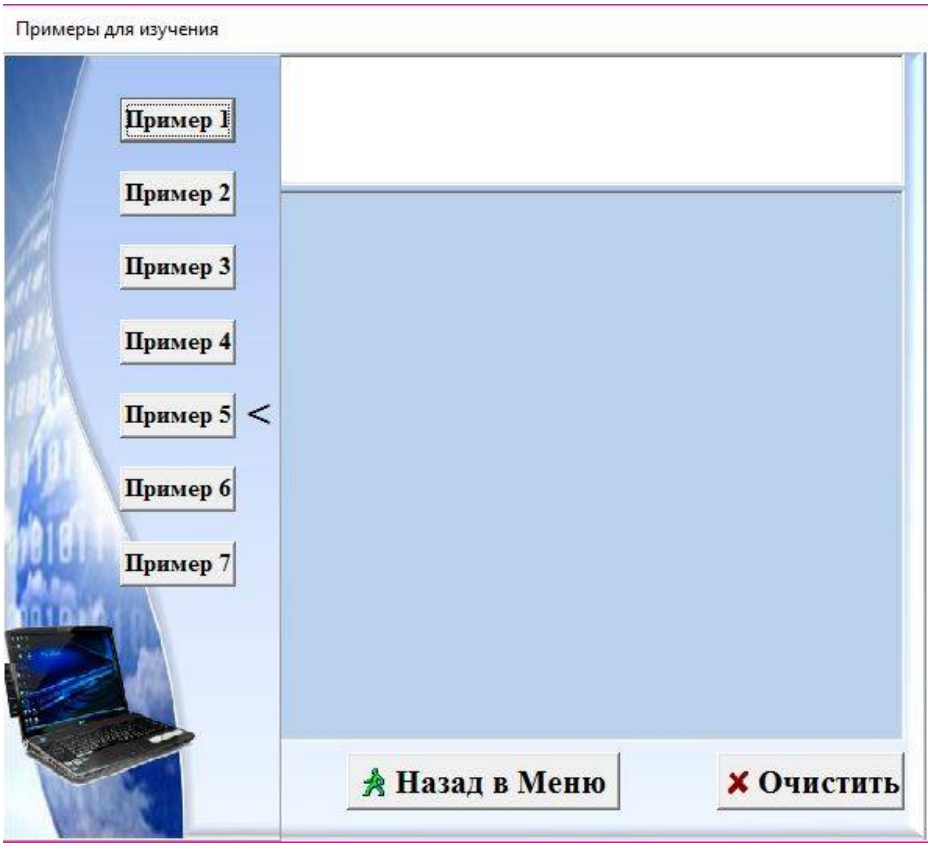

Рис. 2. Примеры для изучения

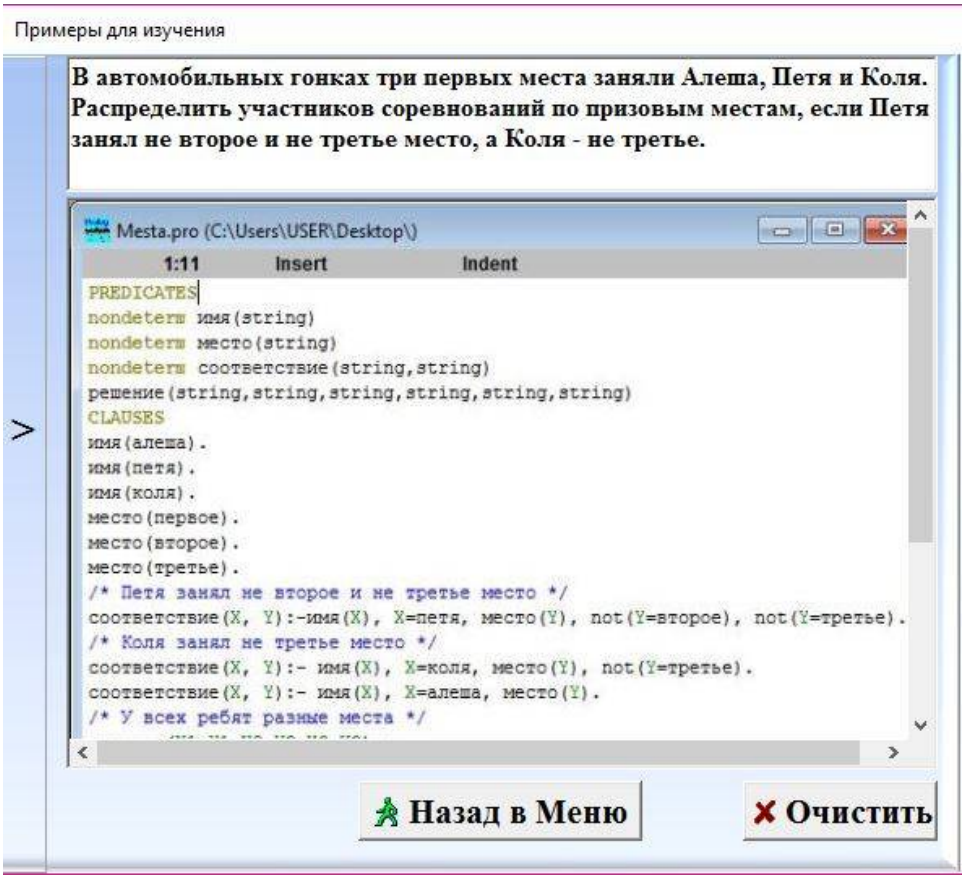

Рис. 3. Примеры для изучения

После изучения типовых примеров становятся доступными сервисы, позволяющие ввести формулировку задачи в текстовом формате (рис.4, рис.5), выявить тип задачи посредством анализа формальных понятий (т.е. посредством выявления в тексте задачи дескрипторов) и сгенерировать соответствующий типовой шаблон Пролог-программы [8].

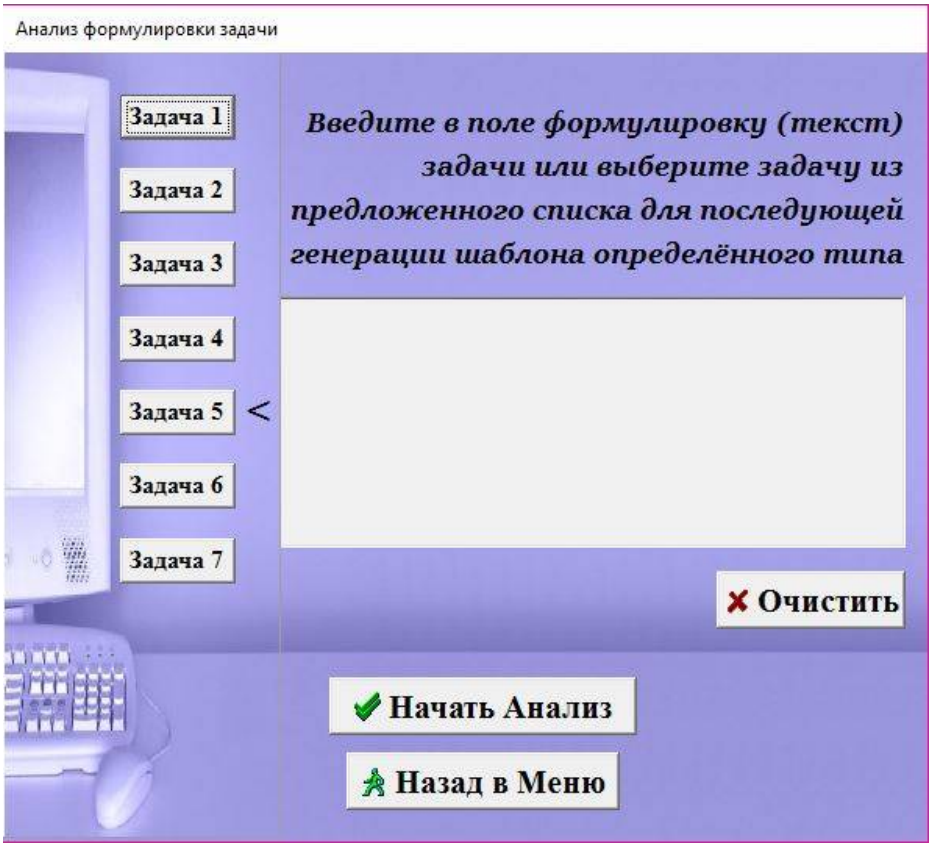

Рис. 4. Выбор задачи для анализа

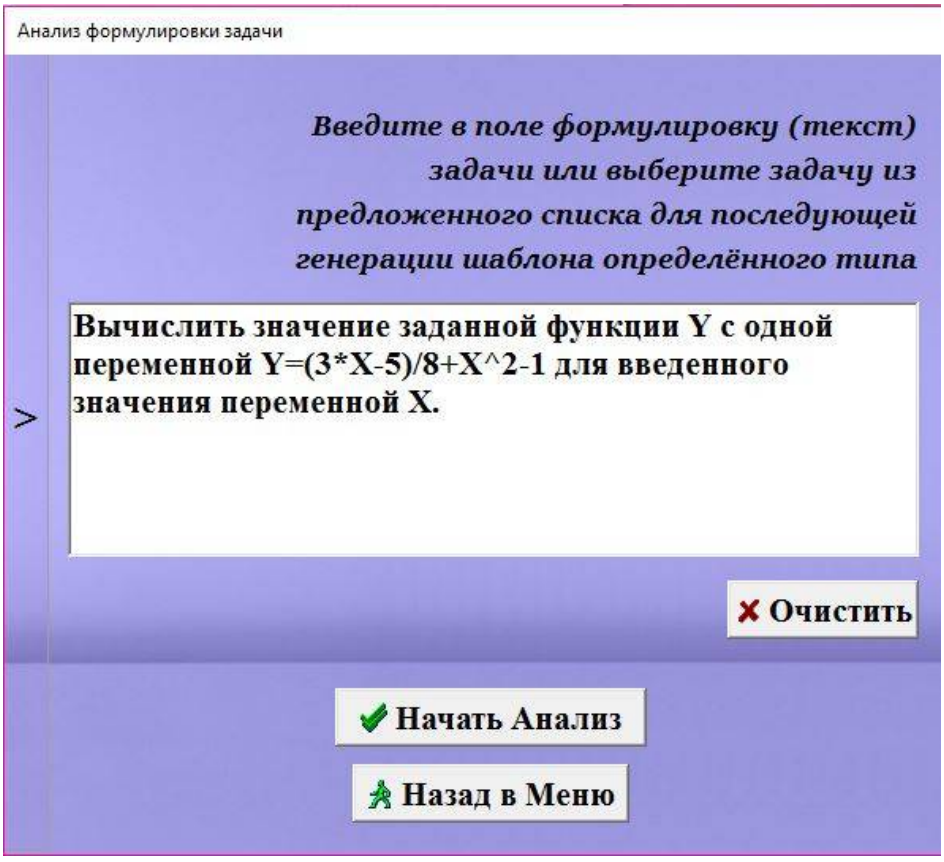

Рис. 5. Анализ формулировки текста

В шаблоне Пролог-программы описание предметной области представлено в форме набора фактов (на основе предикатов различных видов), а также совокупности правил, построенных в соответствии с разработанной методикой (рис.6).

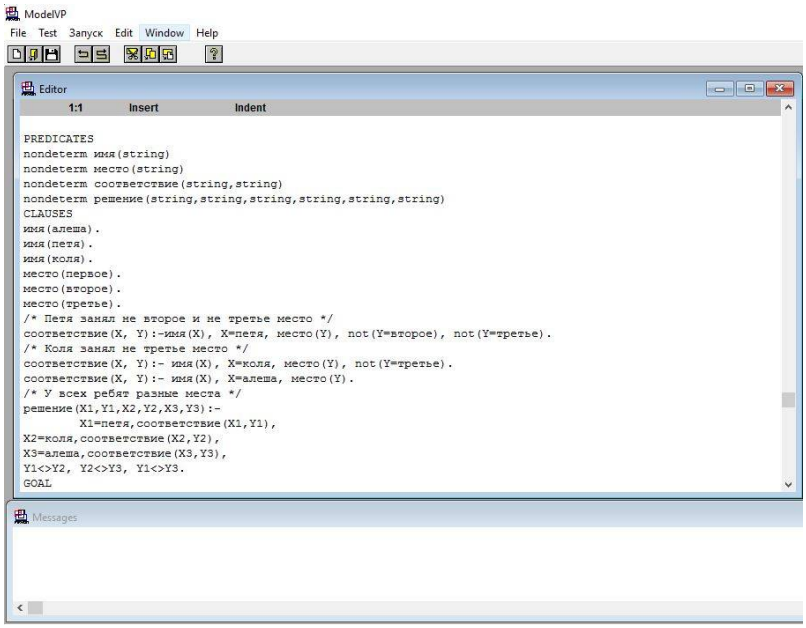

Рис. 6. Окно написания задач, используя типовой шаблон

Для задач первого типа «на выполнение арифметических вычислений» в шаблоне целевая характеристика описывается предикатом с параметрами. Сущность функции, а также ограничения, накладываемые на область определения, описываются в правиле функции. Например, если в формулировке задачи присутствуют дескрипторы «вычислить» и «значение выражения», то в разделе *predicates* следует задать тип объектов предиката в форме: *значение(real,real)*, в разделе *clauses* следует описать правило, которое включает в себя сущность функции, а также ограничения, накладываемые на область определения. В разделе *goal* следует описать целевой предикат, который включает в себя подцель ввода с клавиатуры числовых значений с помощью предикатов *write()/read()*, а также подцель, реализующую обращение к предикату вычисления заданной функции и вывод решения на экран.

При реализации «логических задач» в Пролог-программе конечные множества, между значениями которых устанавливаются взаимно-однозначные соответствия, следует описывать как совокупность фактов, а зависимости между объектами устанавливать с помощью правил. Например, если в формулировке задачи присутствует дескриптор «распределить по местам», то в разделе *predicates* следует задать предикат и тип (домен) объектов предикатов в форме: *соотвеетствие(string,string)*. Это описание означает, что оба объекта предиката *соотвеетствие()* являются строками. В разделе *clauses* описываются факты (причём каждый факт завершается символом «точка») и правила, которые включают в себя ограничения, накладываемые на область определения, а также целевой предикат *решение()*, определяющий заданные формулировкой задачи соответствия. В разделе *goal* осуществляется обращение к целевому предикату и предикату *write()* для вывода решения на экран.

При реализации «задач на анализ значений с последующей выборкой или упорядочением» для нахождения сочетаний и/или перестановок, обусловленных содержанием учебной задачи и правилами комбинаторного анализа, в программе перечисляются факты и описываются правила, а затем объявляется целевой предикат, представляющий собой последовательность соответствий и выражающий смысловое решение задачи. Например, если в формулировке задачи присутствуют дескрипторы «определить» и «высокое/низкое», то в разделе *predicates* следует задать строковый тип (домен) объектов предикатов в форме: *ряд(string,string,string,string,string,string)*. В разделе *clauses* следует описать факты и правила, которые включают в себя ограничения, накладываемые на область определения, а также целевой предикат *ряд()*, определяющий «самый высокий» и «самый низкий» объект. В разделе *goal* с помощью предикатов *write() и read()* необходимо предусмотреть ввод с клавиатуры числовых переменных, после которого будет выполняться обращение к целевому предикату и вывод результата на экран.

В задачах четвёртого типа «на создание баз знаний по предметной области» в Пролог-программе целевая характеристика описывается предикатом с параметрами, которые являются составными объектами. Знания (т.е. истинные утверждения о предметной области) описывается как совокупность фактов, а целью является предикат *write()*, который реализует вывод базы на экран и другие запросы (поиск по критериям). Например, если в формулировке задачи присутствуют дескрипторы вида «создать», «проект», «реализовать», «справочник», «содержится» и «информация», то в разделе программы *domains* следует создать пользовательские домены объектов из базисных типов доменов в форме: имя_переменной=имя_базисного_домена. В разделе *predicates* следует задать тип (домен) объектов предиката в виде:

имя_предиката(имя1_переменной_базисного_домена,

имя2_переменной_базисного_домена),

где переменные имя1_переменной_базисного_домена и

имя2_переменной_базисного_домена будут означать некие совокупности значений. В разделе *clauses* следует описать факты (логические утверждения), которые получены на базе одного и того же целевого предиката *имя_предиката()*, а в разделе *goal* следует осуществить обращение к целевому предикату и предикатам *write() и read()* для вывода базы знаний на экран и организации поиска по введенному с клавиатуры значению.

Далее приведем примеры применения разработанной методики для решения типовых учебных задач [15].

Пример 1. Формулировка задачи. В автомобильных гонках три первых места заняли Алеша, Петя и Коля. Распределить участников соревнований по призовым местам, если Петя занял не второе и не третье место, а Коля - не третье.

Решение (рис.7). Поскольку в формулировке задачи присутствует дескриптор «распределить по местам», то задача является логической. В соответствии с методикой генерации правил представления предметной

области шаблон Пролог-программы должен включать в разделе *predicates* тип объектов предикатов в виде: *соответствие(string,string)* (это описание означает, что оба объекта предиката *соответствие()* являются строками). В разделе *clauses* шаблон должен содержать описание фактов и правил, которые включают в себя ограничения, накладываемые на область определения, а также целевой предикат *решение()*, определяющий правильные соответствия. В разделе *goal* шаблона Пролог-программы должно выполняться обращение к целевому предикату и предикату *write()* для вывода решения на экран. Из этого следует, что шаблон Пролог-программы будет иметь структуру:

PREDICATES

nondeterm имя_предиката1(тип_объекта_предиката1)

nondeterm имя_предиката2 (тип_объекта_предиката2)

…

nondeterm соответствие(тип_объекта_предиката1, тип_объекта_предиката2,…)

решение(тип_объекта_предиката1, тип_объекта_предиката2, тип_объекта_предиката1, тип_объекта_предиката2, тип_объекта_предиката1, тип_объекта_предиката2,…)

CLAUSES

имя_предиката1 (имя_объекта_предиката1).

имя_предиката1 (имя_объекта _предиката2).

…

имя_предиката1 (имя_объекта _предикатаN).

имя_предиката2 (имя_объекта_предиката1).

имя_предиката2 (имя_объекта_предиката2).

…

имя_предиката2 (имя_объекта _предикатаN).

…

имя_предикатаN (имя_объекта_предиката1).

имя_предикатаN(имя_объекта_предиката2).

…

имя_предикатаN (имя_объекта _предикатаN).

соответствие(X, Y):- <Условие1>.

соответствие(X, Y):- <Условие2>.

…

соответствие(X, Y):- <УсловиеN>.

решение(X1,Y1,X2,Y2,…,Xn,Yn):-

X1= имя_объекта_предиката1, соответствие(X1,Y1),

X2= имя_объекта _предиката2, соответствие(X2,Y2),

…

Xn= имя_объекта _предикатаN, соответствие(Xn,Yn),

Y1<>Y2, Y2<>Yn, Y1<>Yn, … .

GOAL

решение(X1,Y1,X2,Y2,…,Xn,Yn),

write(X1,"-",Y1,…,"-",Z1),nl,

write(X2,"-",Y2,…,"-",Z2),nl,

…

write(Xn,"-",Yn,…,"-",Zn),nl.

Далее пользователю (студенту, изучающему логическое программирование) необходимо отредактировать шаблон в соответствии с формулировкой и условием задачи.

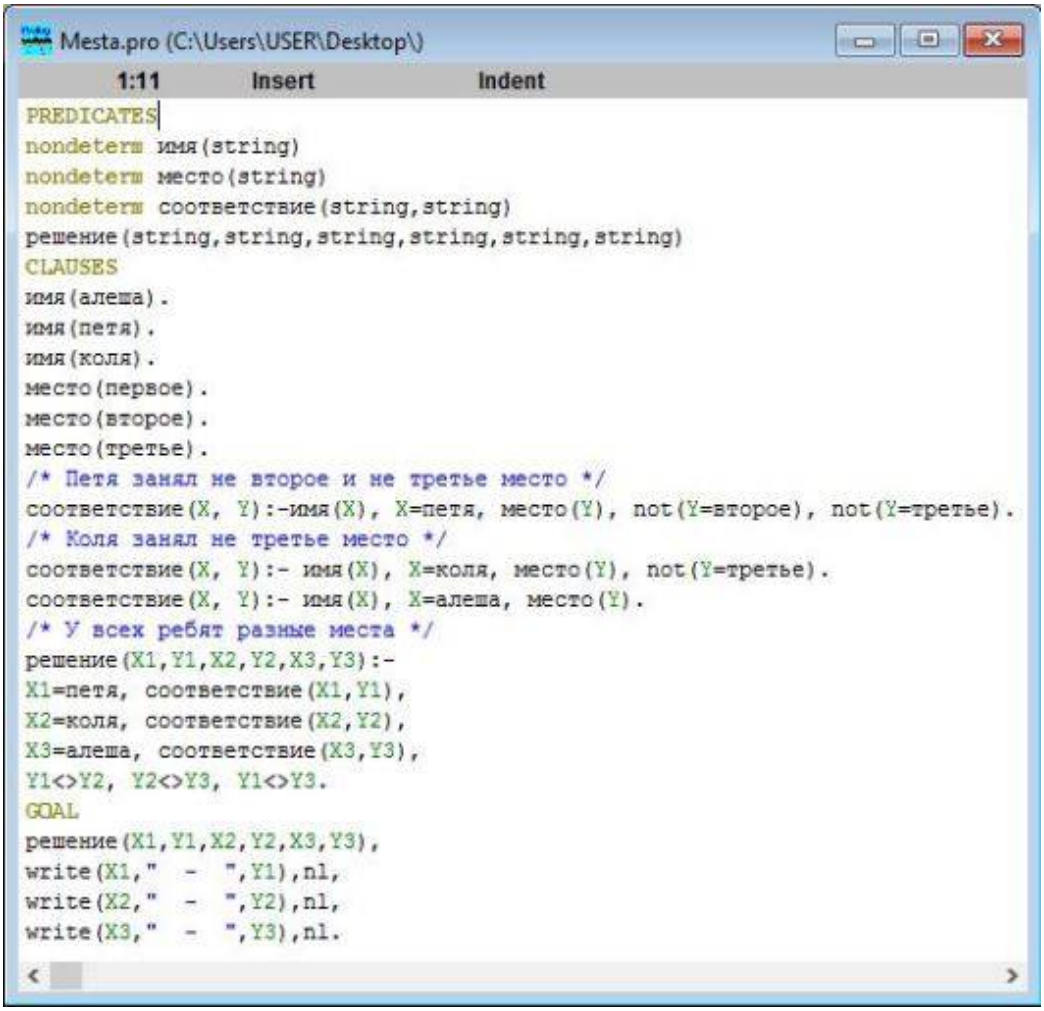

Рис. 7. Демонстрация структуры программы

Пример 2. Формулировка задачи. Наташа, Валя и Аня вышли на прогулку, причем туфли и платье каждой были или белого, или синего, или зеленого цвета. У Наташи были зеленые туфли, а Валя не любит белый цвет. Только у Ани платье и туфли были одного цвета. Установить соответствие цвета туфель и платья каждой из девочек, если у всех туфли и платья были разного цвета.

Решение (рис.8). Поскольку в формулировке задачи присутствует дескриптор «установить соответствие», то тип задачи логический. В разделе *predicates* задаётся домен объектов предикатов в виде: *соответствие(string,string,string)*. В разделе *clauses* описываются факты и правила, которые включают в себя ограничения, накладываемые на область определения, а также целевой предикат *решение()*, определяющий правильные

соответствия. В разделе *goal* выполняется обращение к целевому предикату и предикат *write()* для вывода решения на экран.

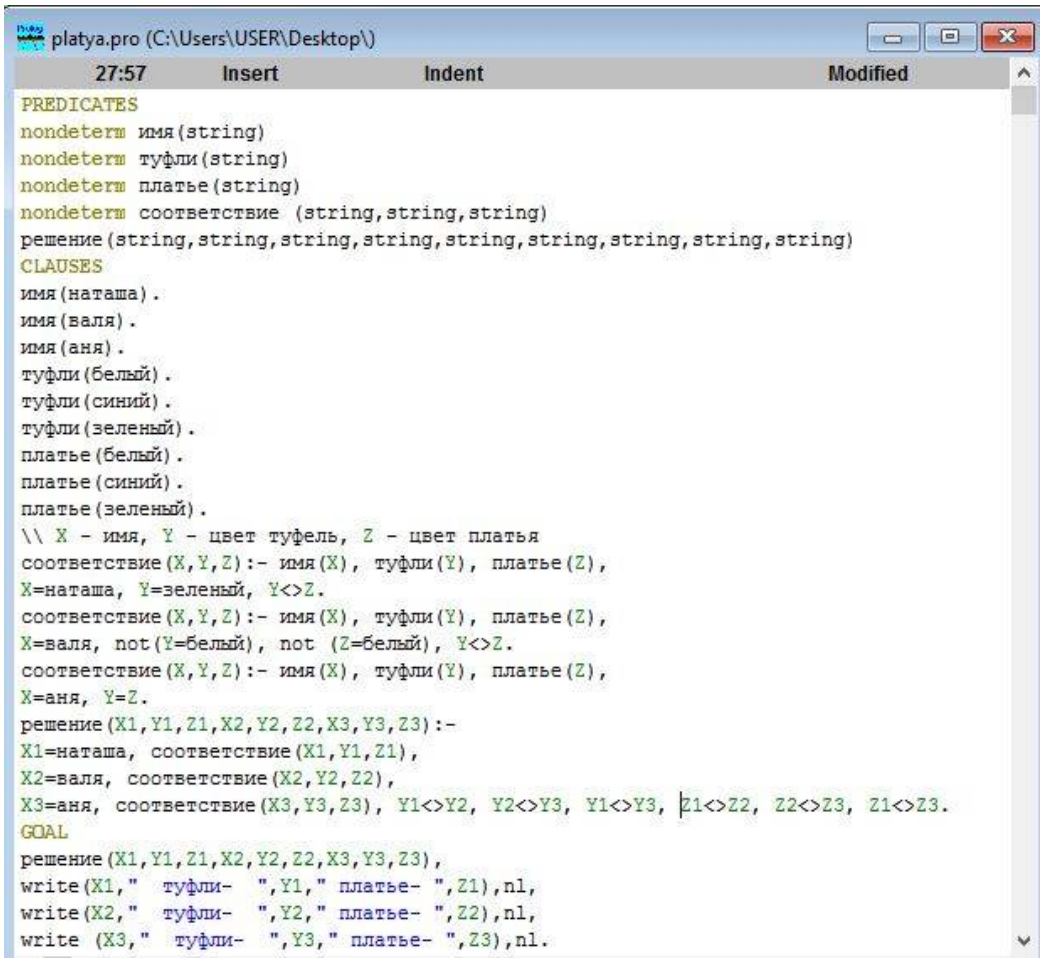

Рис. 8. Демонстрация структуры программы

Пример 3. Формулировка задачи. Вычислить значение заданной функции Z двух переменных Z=(2*X+Y)/(X-Y) для введенных значений переменных X и Y.

Решение (рис.9). Поскольку в формулировке задачи присутствуют дескрипторы «вычислить», то следует определить тип задачи как «задача на выполнение арифметических вычислений». В разделе *predicates* задаётся тип объектов предиката в виде: *значение(real,real)*. Это описание означает, что оба объекта предиката *значение()* относятся к действительным числам. В разделе *clauses* шаблон должен содержать правило, которое включает в себя сущность функции, а также ограничения, накладываемые на область определения. В разделе *goal* шаблона Пролог-программы с помощью предикатов *write() и*

*read()*должен осуществляться ввод с клавиатуры числовых переменных, затем обращение к целевому предикату для вычисления заданной функции и вывода решения на экран. Следовательно, шаблон Пролог–программы будет выглядеть так:

PREDICATES

имя_предиката(тип_объекта_предиката1, тип_объекта_предиката2,…, тип_объекта_предикатаN)

CLAUSES

имя_предиката(X,Y,…,Z):- Если<Условие>,<Арифметические вычисления>;

Иначе<Ограничение>.

GOAL

write("X="), readreal(X),

write("Y="), readreal(Y),

…

write("Z="), readreal(Z), имя_предиката(X,Y,…,Z),nl.

Далее пользователю необходимо отредактировать шаблон в соответствии с формулировкой и условием задачи.

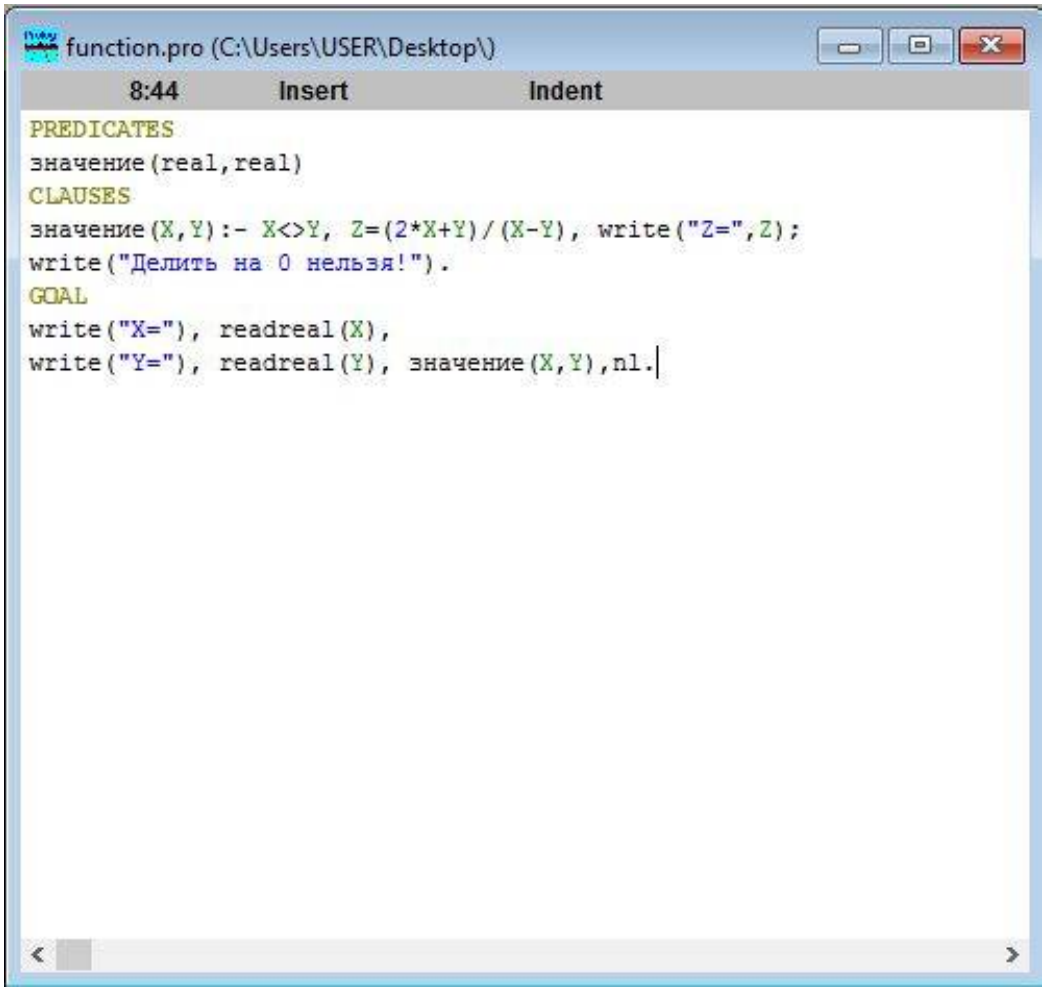

Рис. 9. Демонстрация структуры программы

Пример 4. Формулировка задачи. Вычислить среднее арифметическое для двух введенных чисел.

Решение (рис.10). Поскольку в формулировке задачи присутствует дескриптор «вычислить», то задачу отнесём к типу задач на выполнение арифметических вычислений. В разделе *predicates* задаётся тип объектов предиката в виде: *сред_арифм(integer, integer)*, это описание означает, что оба объекта предиката *сред_арифм()* относятся к целым числам. В разделе *clauses* описывается правило, которое включает в себя математические действия для нахождения среднего арифметического нескольких чисел. В разделе *goal* с помощью предикатов *write() и read()* осуществляется ввод с клавиатуры числовых переменных, после чего выполняется обращение к целевому предикату для вычисления результата и вывода его на экран.

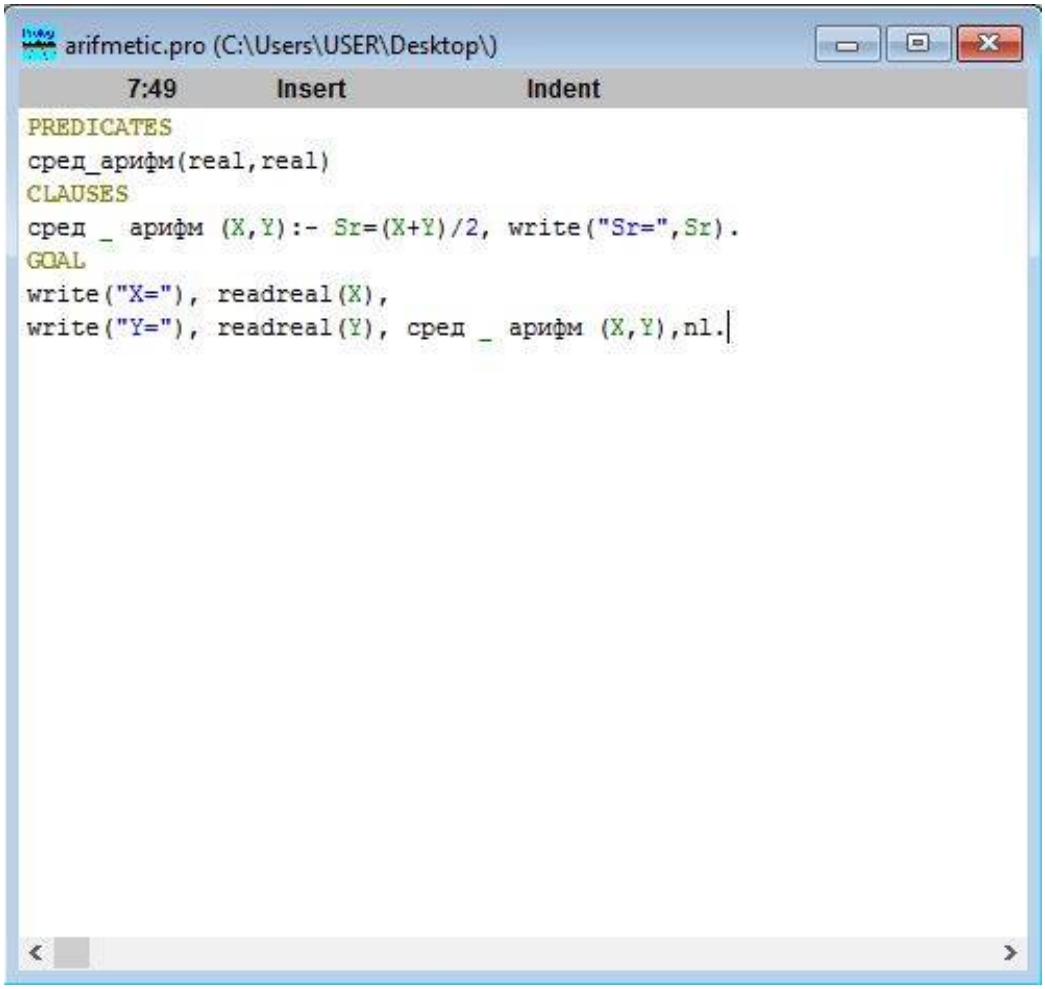

Рис. 10. Демонстрация структуры программы

Пример 5. Формулировка задачи. Определить самое высокое и самое низкое дерево. Если известно, что тополь выше березы, которая выше липы. Клен ниже липы, а сосна выше тополя и ниже ели.

Решение (рис.11). Поскольку в формулировке задачи присутствует дескриптор «определить» и «высокое/низкое», то следует определить тип задачи как «задачи на анализ значений с последующей выборкой или упорядочением». Вразделе*predicates*задаётсятипобъектовпредикатоввиде: *ряд(string,string,string,string,string,string)*. В разделе *clauses* описываются факты и правила, которые включают в себя ограничения, накладываемые на область определения, а также целевой предикат *ряд()*, определяющий самое высокое и самое низкое дерево. В разделе *goal* с помощью предикатов *write() и read()* выполняется ввод с клавиатуры числовых переменных, после чего

осуществляется обращение к целевому предикату и вывод результата на экран. Из этого следует, что шаблон Пролог-программы будет иметь структуру:

PREDICATES

nondetermимя_предиката(тип_объекта_предиката1, тип_объекта_предиката2)

ряд(тип_объекта_предиката1, тип_объекта_предиката2, тип_объекта_предиката1, тип_объекта_предиката2, тип_объекта_предиката1, тип_объекта_предиката2)

CLAUSES

имя_предиката (имя_объекта_предиката1, имя _объекта_предиката2).

имя_предиката (имя_объекта_предиката1, имя _объекта_предиката2).

…

имя_предиката (имя_объекта_предиката1, имя _объекта_предиката2).

ряд(X1,X2,…,Xn):-

выше(X1,X2),выше(X2,X3),…,выше(Xn-1,Xn).

GOAL

ряд(X,Y ,…,Z),write(X,"-",Y,"-",…, "-",Z),nl

Далее пользователю необходимо отредактировать шаблон в соответствии с формулировкой и условием задачи.

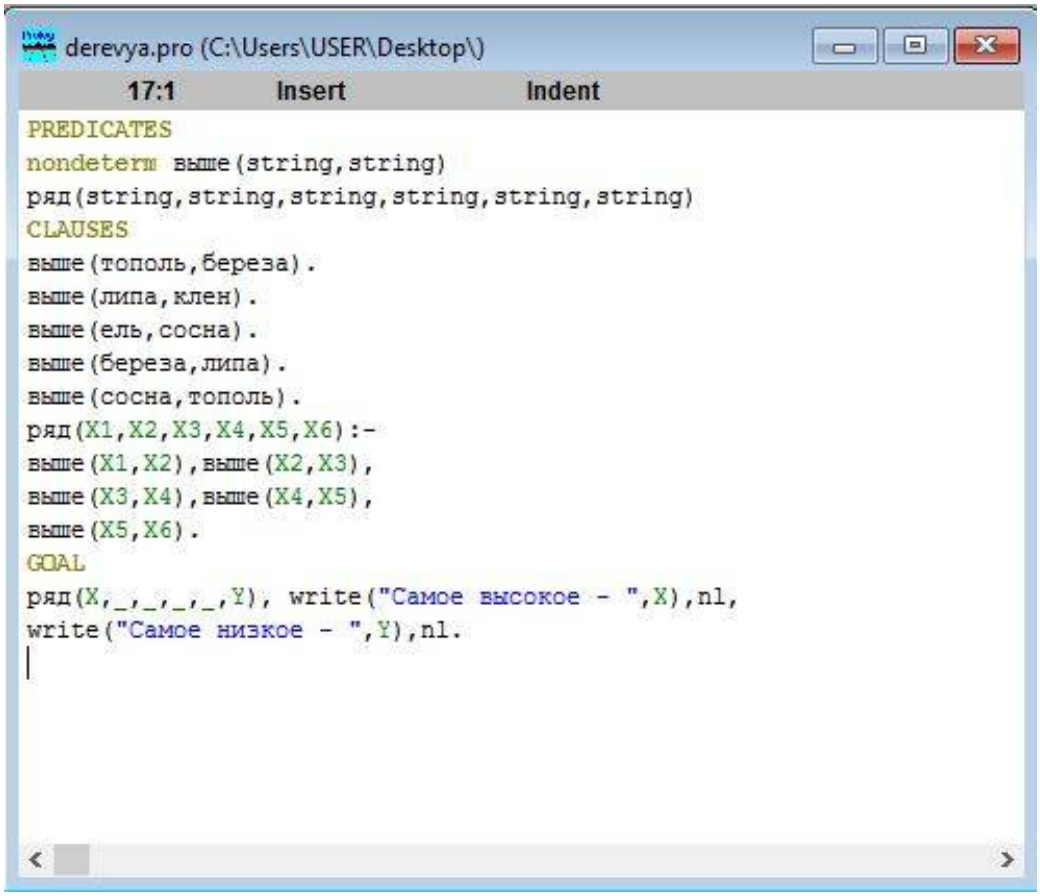

Рис. 11. Демонстрацияструктурыпрограммы

Пример 6. Формулировка задачи. Витя, Юра и Миша сидели на скамейке. Установить в каком порядке они сидели, если известно, что Миша сидел слева от Юры, а Витя слева от Миши.

Решение (рис.12). Поскольку в формулировке задачи присутствует дескриптор «установить порядок», то тип задачи относится к «задачам на анализ значений с последующей выборкой или упорядочением». В разделе *predicates* задаётся тип объектов предикатов в виде: *ряд(string,string,string)*. В разделе *clauses* описываются факты и правила, которые включают в себя ограничения, накладываемые на область определения, а также целевой предикат *ряд()*, определяющий правильный порядок. В разделе *goal* осуществляется обращение к целевому предикату и предикат *write()* для вывода последовательности на экран.

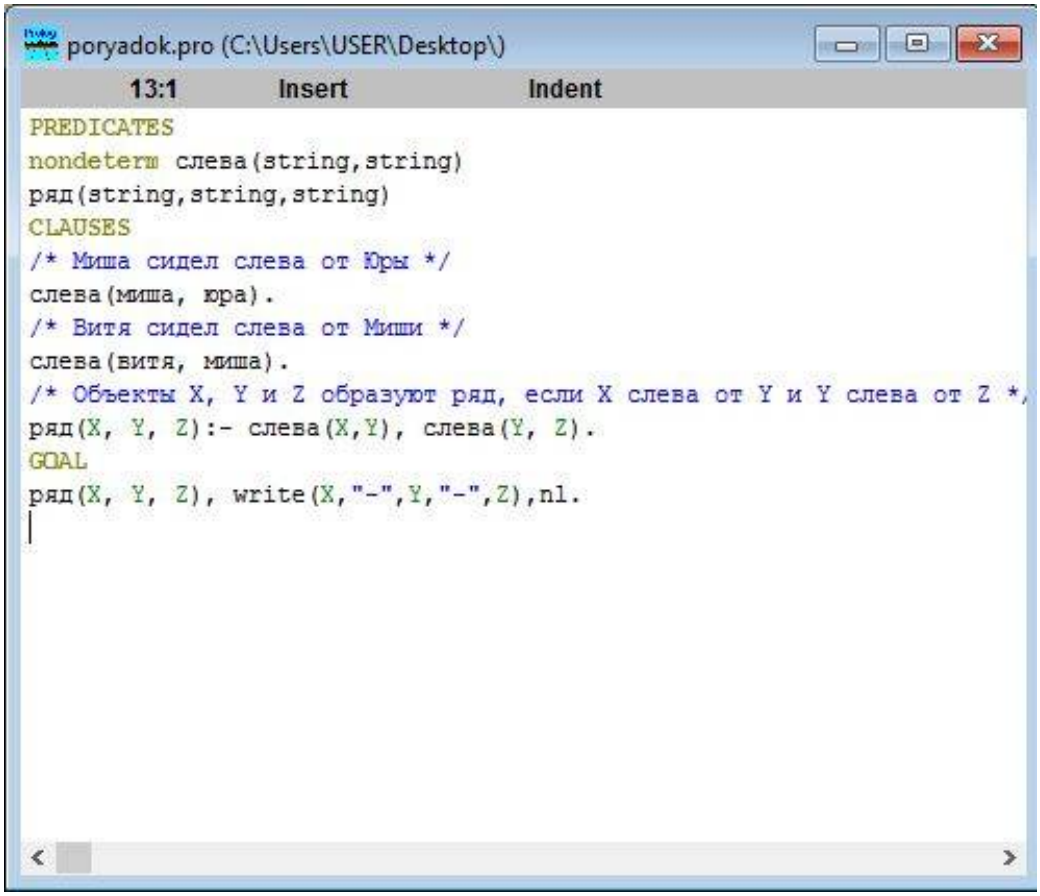

Рис. 12. Демонстрацияструктурыпрограммы

Пример 7. Формулировка задачи. Создать проект, реализующий железнодорожный справочник. В справочнике содержится следующая информация о каждом поезде: номер поезда, пункт назначения и время отправления. Вывести всю информацию из справочника и организовать поиск поезда по пункту назначения.

Решение (рис.13). Поскольку в формулировке задачи присутствуют дескрипторы «создать», «проект», «реализовать», «справочник», «содержится» и «информация», то можно определить тип задачи как «задачи на создание баз знаний по предметной области».

В разделе программы *domains* создадим собственные типы объектов из базисных типов доменов в виде: *nom=integer u punkt, time=string* (которые относятся к целочисленному и строковому типу соответственно). В разделе *predicates* задаётся тип объектов предиката в виде: *poezd(nom,punkt, time)*, это описание означает, что имена *nom,punkt u time* будут означать некие

совокупности (домены) значений. В разделе *clauses* описываются факты (утверждения). Другими словами, все утверждения являются вариациями одного и того же целевого предиката *poezd()*. И, наконец, в разделе *goal* выполняется обращение к целевому предикату и предикат *write()* для вывода базы знаний на экран и предикат *read()* для организации поиска поезда по пункту назначения, введенному с клавиатуры. Таким образом, шаблон Пролог-программы будет выглядеть так:

```
DOMAINS
имя1=тип_домена1
имя2= тип_домена2
имя3= тип_домена3
…
PREDICATES
nondetermимя_базы(имя_1, имя2, имя3,…)
CLAUSES
имя_базы (значение1, значение2, значение3,…).
имя_базы (значение1, значение2, значение3,…).
…
имя_базы (значение1, значение2, значение3,…).
GOAL // вывести всю базу
write("<Заголовок базы>"), nl,
write("имя_1имя_2имя_3… "),
nl, имя_базы (A,B,C,…), write(A,"",B,"",C,"",…),nl,fail.
```

Далее пользователю необходимо отредактировать шаблон в соответствии с формулировкой и условием задачи [8].

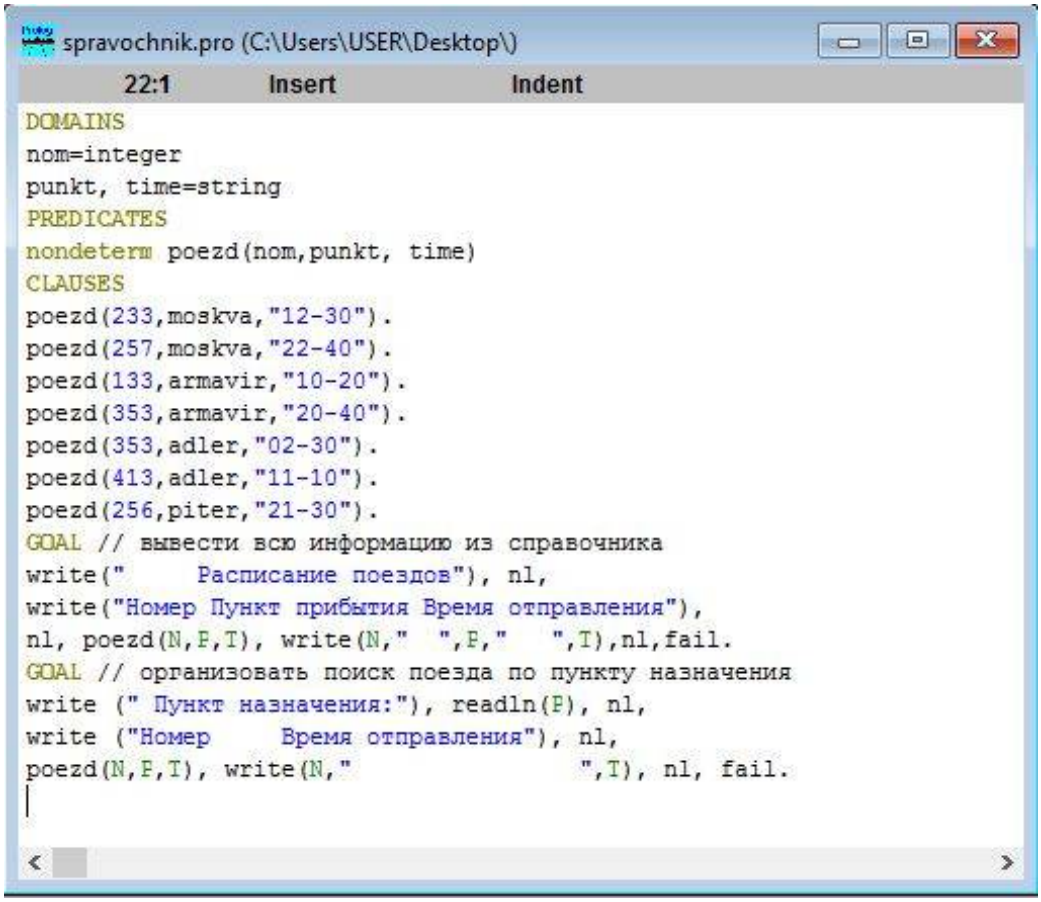

Рис. 13. Демонстрация структуры программы

2.4 Апробация Тренажёра-Практикума

В апробации программного комплекса приняли участие студенты группы Б-41, по мнению которых приложение «Тренажер-практикум» способствует самостоятельному освоению логического программирования на Прологе.

Таким образом, в процессе апробации было опрошено 10 студентов университета, результаты представлены на диаграмме (рис.14).

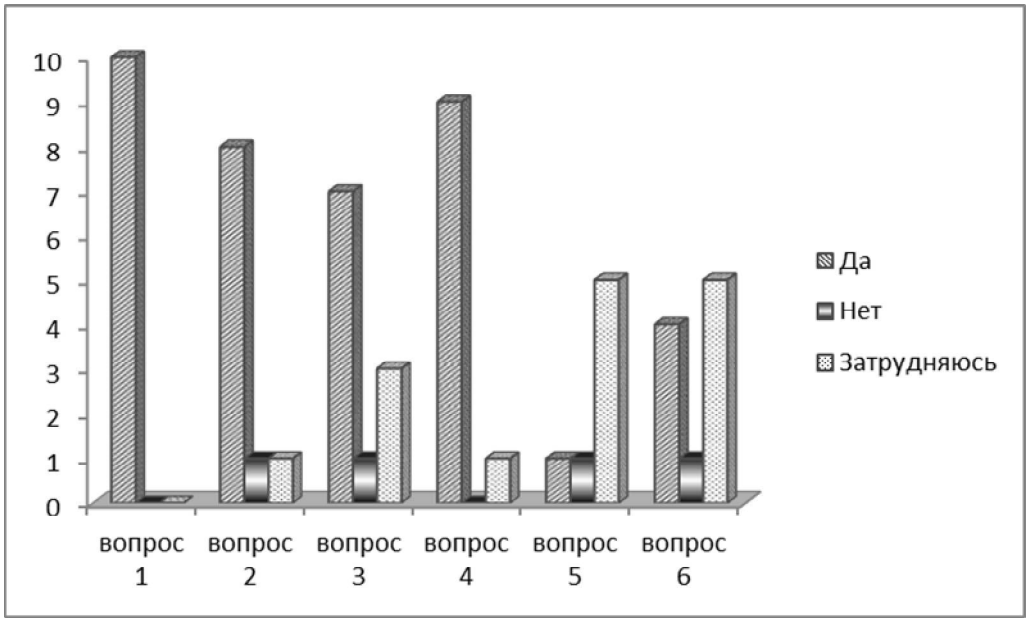

Рис. 14. Результат апробации приложения

Вопросы анкетирования:

1) Понятна ли структура тренажера-практикума?
2) Нравится ли Вам оформление приложения?
3) Удобно ли приложение в использовании?
4) Доступно ли изложен материал в данном тренажере-практикуме?
5) Оцените общую сложность по работе с приложением.
6) Смогли бы Вы с помощью данного тренажера-практикума самостоятельно освоить курс логического программирования?

Заключение

В результате выполнения выпускной квалифицированной работы были решены следующие задачи:

1. Проведён анализ научно-методической литературы в области методов и средств моделирования предметных областей, который позволил обосновать выбор семантических сетей как способа представления знаний при моделировании предметной области в интеллектуальных системах.

2. Разработана типизация учебных задач для языка логического программирования Пролог, включающая следующие типы задач: логические, арифметические, задачи на анализ значений с последующей выборкой или упорядочением и на создание баз знаний по предметной области. При этом для каждого типа задач исследована возможность применения анализа формальных понятий для их решения и составлен комплекс учебных задач.

3. Разработана методика анализа формальных понятий и их группировки в зависимости от параметров объектов предметной области, сущность которой заключается в выявлении инвариантного набора дескрипторов (характеристик), подходящих для представления различных предметных областей, что позволяет реализовать генерацию правил.

4. Разработана методика генерации правил представления предметной области основанная на объявлении предикатов с одним или несколькими параметрами, которые соотнесены с особенностями формулировок типовых учебных задач.

5. Реализована интеллектуальная обучающая система для освоения логического программирования (программный комплекс на языках программирования *ObjectPascal* и *VisualProlog*), обеспечивающая автоматизацию решения учебных логических задач.

Разработанный программный продукт соответствует требованиям технического задания. Апробации программного комплекса показала,

чтоприложение «Тренажер-практикум» способствует самостоятельному освоению логического программирования на Прологе.

Таким образом, все поставленные задачи выполнены; цель достигнута.

Литература

1. Адаменко А.Н., Кучуков А.М. Логическое программирование и VisualProlog. – СПб.: БХВ – Петербург, 2003.
2. Афонин В.А., Макушкин В.А. Интеллектуальные робототехнические системы. // М, 2005.
3. Бобровский С.И. Delphi 7. Учебный курс. С-Пб.: Питер, 2003. 736 с.
4. Боровская Е.В., Давыдова Н.А. Основы искусственного интеллекта: учебное пособие. М.: БИНОМ. Лаборатория знаний, 2010. 127 с.
5. Братко И. Программирование на языке PROLOG для искусственного интеллекта. – М.: Мир, 1990.
6. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. – СПБ.: Питер, 2001.
7. ГоловчинерМ. Н. Введениев системы знаний: курслекций. – Томск, 2011.
8. Давыдова Е.А., Лапенок М.В. Применение анализа формальных понятий для генерации правил представления предметной области при создании учебных программных комплексов // Информационно-коммуникационные технологии в образовании: межвузовский сборник научных работ. Екатеринбург: Урал.гос. пед. ун-т, 2017.
9. Джексон П. Введение в экспертные системы. – М.: Изд. дом «Вильямс», 2001.
10. Джулиан Бакнелл. Фундаментальные алгоритмы и структуры данных в Delphi. – М.: ДиаСофт, 2003. – 560 с.
11. Емельянов Д.А. Практикум по решению задач: введение в Delphi.
12. Информатика: Учебник / Под.ред. Н. В. Макаровой. – М.: Финансы и статистика, 1997.
13. Искусственныйинтеллект. Кн. 1. Системы общения и экспертные системы / Под ред. Э. В. Попова. – М.: Радио и связь, 1990.
14. Искусственныйинтеллект. Кн. 2. Модели и методы / Под ред. Д. А. Поспелова. – М.: Радио и связь, 1990.
15. Козырева Г.Ф. Практикум решения задач по курсу «Основы искусственного интеллекта»:учебно-методическое пособие для студентов, обучающихся по специальности «информатика». Армавир, 2005.
16. Кузнецов С.О. Решетки формальных понятий в современных методах анализа и разработки данных. М.: Поспеловские чтения, 2011. 86 с.

17. ЛюгерДж.Ф. Искусственный интеллект: стратегии и методы решения сложных проблем. – М.: "Вильямс", 2003.
18. Макаренко С. И. Интеллектуальные информационные системы: учебное пособие. – Ставрополь: СФ МГГУ им. М. А. Шолохова, 2009. – 206 с.: ил.
19. Материалы Российской ассоциации искусственного интеллекта // http://www.raai.org/resurs/resurs.shtml?ilinks.
20. Николаев А.Б. Интеллектуальные системы: учебное пособие / А.Б. Николаев, А.В. Остроух - М.:МАДИ, 2012. – 271 с.: ил.
21. Печень О.А. Интеллектуальныеинформационные системы: Курс лекций. Новосибирск, 2005.
22. Попов Э.В.Справочник по искусственному интеллекту: в 3-х т. // Под ред. Попова Э.В. и Поспелова Д.А. // – М.: Радио и связь, 1990.
23. Представление и использование знаний/ Под ред. Х. Уэно, М. Исидзука. – М.: Мир, 1987.
24. Пупков К. А., Коньков В. Г. Интеллектуальные системы. – Изд-во МГТУ им. Н. Э. Баумана, 2003.
25. Рассел С. Искусственный интеллект: современный подход / С. Рассел, П. Норвинг ; пер. с англ. – М: Издат. дом «Вильямс», 2006. 1408 с.
26. Федоров А.Г. Создание Windows – приложений в среде Delphi. М.: Компьютер Пресс, 1995. 287 с.
27. Чинакал В.О. Интеллектуальные системы и технологии: Учеб.пособие. – М.: РУДН, 2008. – 303 с.: ил.
28. ЯсницкийЛ. Н. Введение в искусственный интеллект. – Пермь: Изд-во Пермского университета, 2001. 143 с.
29. ЯсницкийЛ. Н. Интеллектуальные информационные технологии и системы. – Пермь: Пермский государственный университет, 2007. 271 с.
30. Ясницкий Л.Н. Интеллектуальные системы: учебник. М.: Лаборатория знаний, 2016. 221 с.

Приложения

Приложение 1.

Фрагмент программы на языке ObjectPascal.

```
unitProlog;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls, jpeg;
type
  TForm1 = class(TForm)
    Panel1: TPanel;
    Label1: TLabel;
    Button1: TButton;
    Button2: TButton;
    Button3: TButton;
    Button4: TButton;
    Button5: TButton;
    Label2: TLabel;
    Timer1: TTimer;
    Image1: TImage;
    Image2: TImage;
procedureFormCreate(Sender: TObject);
procedure Label2Click(Sender: TObject);
procedure Timer1Timer(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
private
{ Private declarations }
public
{ Public declarations }
end;
var  Form1: TForm1; b:boolean;
```

```
implementation
uses Prolog2, Prolog3;
{$R *.dfm}
procedure TForm1.FormCreate (Sender: TObject);
begin
b:=false;
Application.HintColor:=clHighlight;
end;
        procedure TForm1.Label2Click(Sender: TObject);
        begin
        b:=not(b);
        if b then Label2.Caption:='<'
        else Label2.Caption:='>';
        end;
procedure TForm1.Timer1Timer(Sender: TObject);
begin
if (b) and (Panel1.Left<0) then
Panel1.Left:=Panel1.Left+5;
if (not b) and (Panel1.Left>-245) then
Panel1.Left:=Panel1.Left-5;
end;
        procedure TForm1.Button1Click(Sender: TObject);
        begin
        Form2.Show;
        end;
        procedure TForm1.Button2Click(Sender: TObject);
        begin
        Form3.Show;
        end;
procedure TForm1.Button5Click(Sender: TObject);
begin
Form1.Close;
end;
end.
```

Приложение 2.

Фрагмент программы на языке VisualProlog.

```
%BEGIN Арифметическиевычисления, InitControls, 18:51:24-12.3.2017, Code automatically updated!
win_CreateControl(wc_PushButton,rct(80,65,260,115),"Задача 4",_Win,[wsf_Group,wsf_TabStop],idc_задача_4),
win_CreateControl(wc_PushButton,rct(80,130,260,180),"Задача 5",_Win,[wsf_Group,wsf_TabStop],idc_задача_5),
win_CreateControl(wc_PushButton,rct(80,195,260,245),"Задача 6",_Win,[wsf_Group,wsf_TabStop],idc_задача_6),
%END Арифметическиевычисления, InitControls
%BEGIN Task Window, id_Test_editir_windows
task_win_eh(_Win,e_Menu(id_Test_editir_windows,_ShiftCtlAlt),0):-!,
win_editor_Create(_Win),
!.
%END Task Window, id_Test_editir_windows
%BEGIN TaskWindow, id_Запуск_Примеры_арифметические_вычисления
task_win_eh(_Win,e_Menu(id_Запуск_Примеры_арифметические_вычисления,_ShiftCtlAlt),0):-!,
win_арифметические_вычисления_Create(_Win),
!.
%END TaskWindow, id_Запуск_Примеры_арифметические_вычисления
%BEGIN TaskWindow, id_Запуск_Примеры_бз_по_предметной_области
task_win_eh(_Win,e_Menu(id_Запуск_Примеры_бз_по_предметной_области,_ShiftCtlAlt),0):-!,
win_бз_по_предметной_области_Create(_Win),
!.
%END TaskWindow, id_Запуск_Примеры_бз_по_предметной_области
%BEGIN TaskWindow, id_Запуск_Примеры_логические_задачи
task_win_eh(_Win,e_Menu(id_Запуск_Примеры_логические_задачи,_ShiftCtlAlt),0):-!,
win_логические_задачи_Create(_Win),
!.
%END TaskWindow, id_Запуск_Примеры_логические_задачи
```

```
        %BEGIN Task Window, e_Activate
task_win_eh(_Win,e_Activate,0):-!,
        !.
        %END Task Window, e_Activate
        %BEGIN Task Window, id_help_contents
task_win_eh(_Win,e_Menu(id_help_contents,_ShiftCtlAlt),0):-!,
        vpi_ShowHelp("modelvp.hlp"),
        !.
        %END Task Window, id_help_contents
predicates
win_editor_eh : EHANDLER
clauses
win_editor_Create(_Parent):-
ifdefuse_editor
        file_str ("F:\\ДИПЛОМ\\ModelVP\\Z1.pro",Text),
        Font = font_Create(ff_Fixed,[],10),
        ReadOnly = b_false, Indent = b_true, InitPos = 1,
        edit_Create(win_editor_WinType,win_editor_RCT,win_editor_Title,
                win_editor_Menu,_Parent,win_editor_Flags,Font,ReadOnly,
                Indent,Text,InitPos,win_editor_eh),
enddef
        true.
```