МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«**Вятский государственный университет**»

Факультет автоматики и вычислительной техники

Кафедра электронных вычислительных машин

Отчет

Комплекс лабораторных работ по дисциплине

«Проектирование информационных систем»

Выполнил студент группы ИВТ-42 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Щесняк Д.С./

Проверил доцент кафедры ЭВМ \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/Караваева О.В./

Киров 2018

# **Введение**

Обработка знаний – одна из сфер применения искусственного интеллекта, которая предполагает использование компьютером знаний для решения проблем и принятия решений. Переход от архитектуры, которая ориентирована на обработку данных, к архитектуре обработки знаний, является одной из ключевых проблем развития ЭВМ. Важную роль в обработке знаний играет логический вывод. С реализацией логического вывода связано три проблемы: разработка методов логического вывода, создание ПО для описания и решения поставленной задачи логическим выводом, аппаратная поддержка логического вывода.

В настоящее время реализовано множество программ логического вывода. В своей основе данные программы для решения задач используют метод резолюции. Существует другой метод, который за меньшее количество шагов позволяет выполнить логический вывод. Данный метод использует операцию деления дизъюнктов.

Наибольший интерес представляет логический вывод в логике предикатов первого порядка. Данный аппарат математической логики имеет хорошо изученную теоретическую базу и представляет большие возможности по сравнению с логикой высказываний.

Таким образом, разработка программного обеспечения логического вывода путем применения метода обобщенного деления дизъюнктов является перспективный направлением для развития данной области. Программная реализация позволит более детально изучить данный метод и рассмотреть возможные способы его развития.

# **Анализ предметной области**

В данной главе производится анализ предметной области, производится обзор существующих программных решений поставленной задачи.

## **Анализ состояний проблемы**

Исчисление высказываний ограничено структурой предложений в терминах простых высказываний, которых покрывают лишь малую часть высказываний. Исчисление предикатов вводит предикаты, которые позволяют строить высказывания вида: Джон любит вино.

Из высказываний составляется база знаний, которая используется в логическом выводе. Всего существует 3 основных вида логического вывода:

1. Дедукция. Дедукция предполагает выведение результата из общих правил и посылок.
2. Индукция. Предполагает выведение общего правила из заданных предпосылок и результата.
3. Абдукция – процесс выведения предпосылки из общего правила и результата.

Данные логические выводы можно наглядно проиллюстрировать на известном силлогизме:

1. Все люди смертны. Ɐ x Человек(х) => Смертен(x)
2. Сократ является человеком. Человек(Сократ)
3. Сократ смертен. Смертен(Сократ)

В данном случае первое высказывание является правилом, второе – посылкой, а третье – выводом. Дедукция позволяет на основе высказываний, что все люди смертны и Сократ – человек, сделать вывод, что Сократ тоже смертен. Индукция же на основе того, что Сократ смертен и является человеком, позволяет сделать вывод, что все люди смертны. Абдукция выводит посылку о том, что Сократ человек на основе высказываний о смертности людей и Сократа.

Все методы логического вывода могут быть классифицированы по следующим критериям:

* По форме представления данных.
* По законам логики.
* По принципу логического вывода.
* По направленности. Методы характеризуются направлением вывода: от заключения к исходным посылкам, от исходных посылок к заключению, либо двунаправленный.
* По тактике обхода. Определяет по какому правилу будет обходиться дерево логического вывода.
* По стратегии. Выбор правильной стратегии позволяет повысить эффективность поиска необходимых дизъюнктов.

Классификация методов логического вывода представлена на рисунке 1.1.

В настоящее время существует множество методов логического вывода в логике предикатов. Большинство из них используют метод резолюций, который последовательно производит логический вывод, из-за чего процесс логического вывода может занять время.

Одним из наиболее перспективных методов, которые могут ускорить логический вывод является метод обобщенного деления дизъюнктов, в основе которого лежит стратегия вывода вширь, благодаря которой возможно делать параллельный вывод.



Рисунок 1.1 – Классификация методов логического вывода

## **Обзор аналогов**

На данный момент имеется очень малое количество информации о программах логического вывода, использующих в своей основе метод деления дизъюнктов. Найденные программы были разработаны на кафедре ЭВМ ВятГУ и осуществляют логический вывод только в логике высказываний.

**Логический вывод в исчислений высказываний**

Данная программа была разработана на кафедре ЭВМ и представляет собой инструмент для логического вывода в исчислений высказываний.

Программа представляет собой приложение разработанное для операционной системы Windows, которая имеет следующие функциональные возможности:

* Дедуктивный и абдуктивный логический вывод методом деления дизъюнктов, а так же их модификации.
* Детальный отчет о логическом выводе с разбитием по шагам
* Преобразование исходных высказываний в вид дизъюнктов с последующим логическим выводом

Большой выбор методов логического вывода позволяет решать широкий круг задач. Однако функционал программы позволяет делать логический вывод только в логике предикатов, что существенно ограничивает множество доступных высказываний. Также существует ряд следующих недостатков:

* Отсутствие возможности ввода нескольких выводимых высказываний
* Отсутствие интерфейса, позволяющего редактировать набор высказываний загруженных из файла
* Игнорирование ошибок ввода исходных высказываний, что влечет за собой некорректную работу программы.

Пример графического интерфейса приложения представлен на рисунке 1.1. Пример выполнения абдуктивного логического вывода представлен на рисунке 1.2.

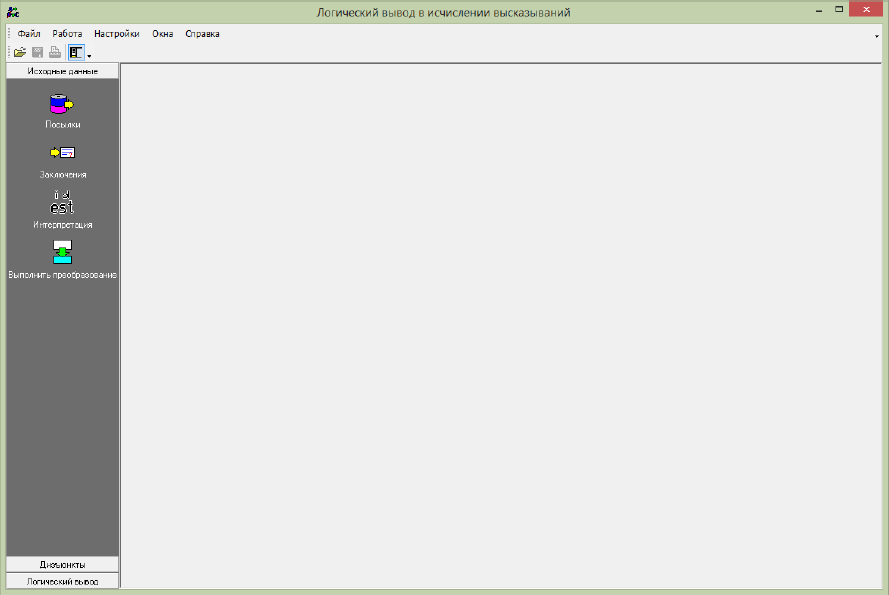


Рисунок 1.1 – Графический интерфейс программы логического вывода

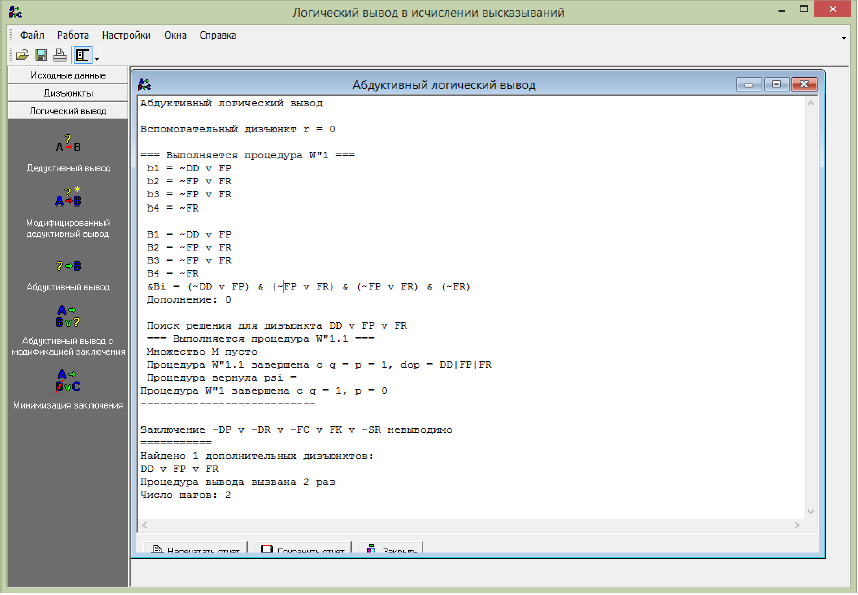


Рисунок 1.2 – Пример выполнения абдуктивного логического вывода

**Lince**

Данная программа была разработана в ходе дипломного проектирования студентами кафедры ЭВМ.

Программа представляет собой приложение разработанное на языке программирование Java для операционных систем семейства Windows, которая имеет следующие возможности:

* Логический вывод в логике высказываний методом деления дизъюнктов.
* Построение графической схемы вывода высказываний.
* Поддержка вывода нескольких высказываний.
* Дедуктивный и абдуктивный логический вывод.
* Вывод информации о выполняемых процедурах в журнал выполнения

Основным недостатком данной программы является логический вывод только в логике высказываний, что существенно ограничивает множество возможных высказываний.

Пример графического интерфейса представлен на рисунке 1.3.

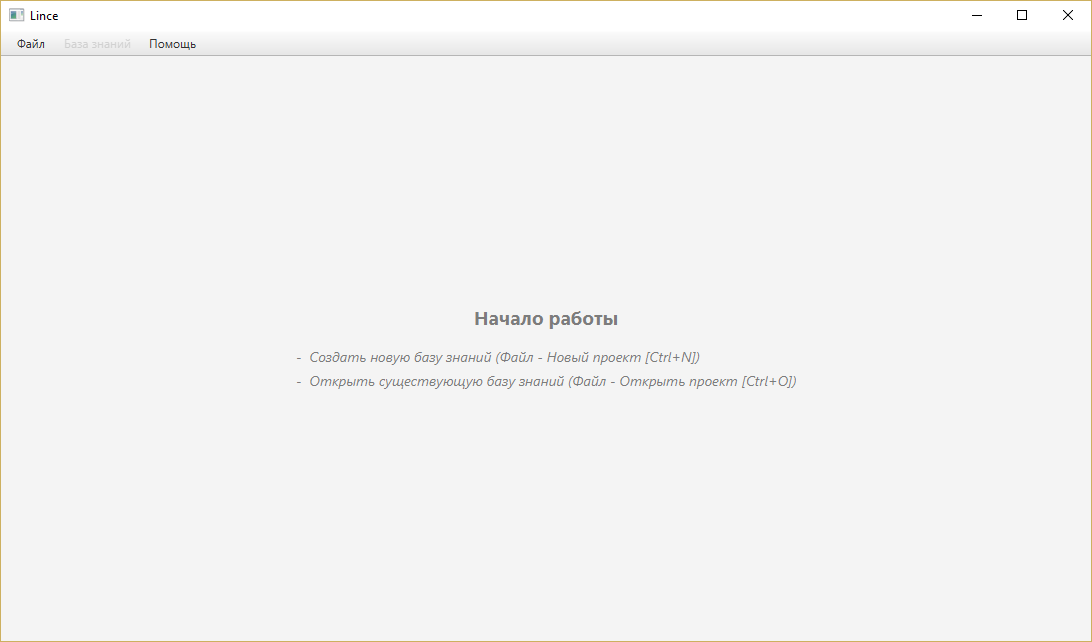


Рисунок 1.3 – Графический интерфейс приложения Lince

## Анализ систем программирования

**Visual prolog**

Является объектно-ориентированной модификацией пролога со строгой типизацией. Логические языки программирования традиционно являются интерпретируемыми, но visual prolog является компилируемым, благодаря чему появляется возможность отлавливания синтаксических ошибок на стадии компиляции. На фоне остальных реализация выделяется возможностью удобного проектирования пользовательского графического интерфейса. Основным недостатком данной реализации является доступность только под операционной системой windows. Пример графического интерфейса представлен на рисунке 1.4.

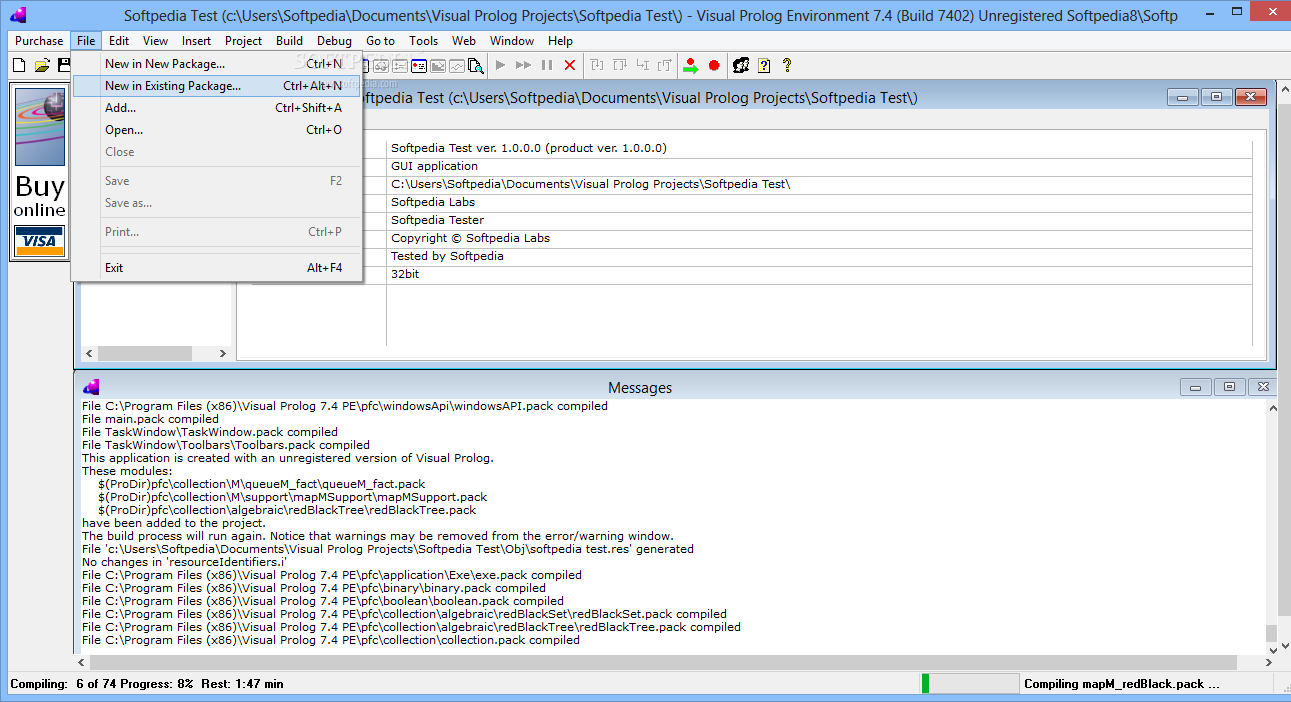


Рисунок 1.4 – Графический интерфейс Visual Prolog

**Swi-prolog**

Swi-prolog является свободной реализацией языка программирования пролог. Имеет богатый набор функций, библиотек для реализации параллельных вычислений, разработки графического интерфейса, интегрирования с другими языками программирования, в том числе и С++. Поддерживает современные операционные системы, такие как Linux, windows, macintosh.

SWI-Prolog включает в себя быстрый компилятор, профилировщик, набор библиотек и удобный интерфейс для подключения C-модулей. Он реализован для ряда UNIX-платформ, таких, как HP, IBM Linux, для NeXT, OS/2, Sun и Sparc

Пример графического интерфейса представлен на рисунке 1.5

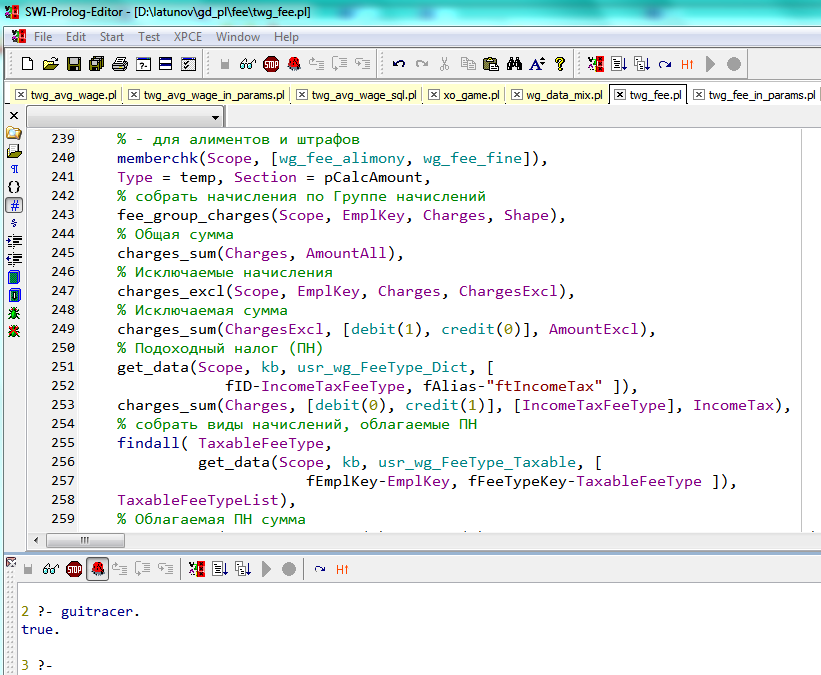


Рисунок 1.5 – Графический интерфейс SWI-prolog

## Выводы

Проведя анализ предметной области и рассмотрев программные решения по заданной проблеме можно сделать выделить следующие тенденции:

* Большинство существующих методов логического вывода опираются метод резолюций, который из-за последовательного выполнения не может быть оптимально распараллелен на современных ЭВМ.
* Существует небольшое количество программных решений реализующий метод деления дизъюнктов и отсутствие программ реализующих логический вывод в логике предикатов первого порядка.
* Проблема представления знаний имеет обширную теоретическую и практическую базу.

# Техническое обоснование проекта. Задачи курсового проектирования

В данном разделе, на основе анализа аналогичных программных решений выдвигаются требования к функциональным характеристикам, к надежности, к графическому интерфейсу, которым должно удовлетворять приложение. Также приводится обоснование необходимости разработки и назначение программы.

## Обоснование необходимости разработки

В настоящее время переход от обработки данных к обработке знаний является одним из приоритетных и перспективных направлений разработки искусственного интеллекта. Одним из главных направлений данного течения является логический вывод, который имеет хорошую базу методов.

Программы, рассмотренные в пункте 1.2, имеют существенный недостаток в виде возможности логического вывода только в логике высказываний. Модификация данных программ затруднена изменением их архитектуры. В связи с этим принято решение разработать собственную программу логического вывода в логике предикатов первого порядка методом деления дизъюнктов с использованием современных средств разработки программного обеспечения.

## Постановка задачи на проектирование

Разрабатываемая программа предназначена для научного исследования логического вывода методом деления дизъюнктов в логике предикатов первого порядка. Основная цель данной программы разработка практического инструмента реализующий теоретическую базу метода деления дизъюнктов. Данная программа на целена на демонстрацию возможностей данного метода, а так же для практического применения как студентами так и преподавателями.

## **Функциональные характеристики**

Разрабатываемая программа должна обладать следующими характеристиками:

* Реализация дедуктивного метода логического вывода в логике предикатов первого порядка.
* Обработка исходных данных, которыми являются набор фактов, высказываний, выводимых высказываний, которые представлены в виде последовательности дизъюнкций – предикатов, разделенных дизъюнкцией.
* Возможность просмотра логов процесса вывода
* Сохранение в виде файла и загрузка базы знаний и всех сопутствующих пользовательских настроек
* Реализация меню настроек, внутри которых возможно менять вид вводимых констант, изменение логики редактора высказываний.

## Требования к программной совместимости

Разрабатываемое программное обеспечение должно работать на всех современных операционных систем для рабочего стола: Windows 7/8/10, Mac OS X, Ubuntu 16.

Также разрабатываемая программа должна одинаково выглядеть на всех современных операционных системах.

## **Требования к надежности**

Надежное функционирование приложения должно быть обеспечено выполнением конечным пользователем ряда требований по эксплуатации:

* Организация стабильной работы вычислительного устройства, в данном случае – персонального компьютера.
* Обеспечение достаточных вычислительных мощностей описанных в пункте 2.6.

Со стороны разработчика приложение гарантирует корректную обработку всех входных данных, и если они не соответствуют желаемым, то оповещение об этом пользователя.

## **Условия эксплуатации**

Минимальные системные требования, которые необходимы для корректной работы программного обеспечения:

* 32-х разрядный процессор с тактовой частотой 1 Ггц или выше
* 1 Гб ОЗУ
* 30 Мб свободного места на жестком диске

## **Требования к исходным кодам и языкам программирования**

Программное обеспечение должно быть разработано на С++ для достижения наибольшей производительности.

## Требования к программной документации

Разрабатываемое приложение будет поставляться со следующей программной документацией:

* Руководство пользователя, которое включает в себя описание всех возможностей приложения и исчерпывающую информацию по назначению каждого элемента.
* Исходный код с документацией.

## Требования к интерфейсу

Интерфейс программы должен быть интуитивно понятным и наиболее простым для конечного пользователя. В процессе использование программы, пользователь должен получать необходимые сообщения об ошибках, предупреждений для того, чтобы правильно реагировать на состояние программы.

В общем случае интерфейс должен состоять и следующих элементов:

* Главное меню программы состоящее из создания нового проекта, открытия существующего, сохранения проекта, меню редактирования, внутри которого возможна смена шрифта, отмена последнего действия в редакторе высказываний и тд. Меню настроек, имеющее возможность подстроить программу под нужны пользователя. Меню помощи, которое будет содержать пользовательскую документацию.
* Окно настроек, в котором будут находится пункты меню установленные заказчиком: начало констант с прописных и строчных символов, автоподстановка истинной импликации.
* Окно редактирования высказываний, содержащее редактор высказываний, окно логов.

# Структурный подход

Для проведения анализа и реорганизации бизнес-процессов предназначено CASE-средство AllFusion ERwin Process Modeler, поддерживающее методологии: IDEF0 (функциональная модель), IDEF3 (Workflow Diagram), DFD (DataFlow Diagram).

Построение модели системы начинается с описания функционирования системы в целом.

## Описание системы

Перед запуском логического вывода пользователю необходимо ввести исходные и выводимые дизъюнкты и дизъюнкты, на основе этих данных будет производится логический вывод. Все исходные дизъюнкты представляют из себя предикаты, разделенные «логическим или», записанные в сколемовской форме.

Далее, на основе алгоритма логического вывода обобщенным делением дизъюнктов, производится попытка вывести выводимое высказывание через исходные путем различных подстановок переменных. Если логический вывод успешно завершился, то выдается множество подстановок, являющиеся решением, в противном случае множество является пустым.

Для более подробного рассмотрения пошагового вывода, выполняется построение дерева логического вывода.

Контекстная диаграмма IDEF0 представлена на рисунке 3.1

Функционирование системы описывается в терминах:

* вход:
  + Исходные дизъюнкты
  + Выводимые дизъюнкты
* выход
  + Дерево логического вывода
  + Результирующие подстановки
* управление
  + Регламент
  + Алгоритм логического вывода
* механизм
  + Пользователь;
  + Система.



Рисунок 3.1 – Контекстная диаграмма IDEF0

Логический вывод состоит из 4 основных этапов:

* Получение исходных данных. Ввод пользователем исходных дизъюнктов, которые будут составлять базу знаний и выводимый дизъюнкты
* Обработка исходных данных и составление из них базы знаний при помощи лексического и синтаксического анализа
* Произведение логического вывода по заданному алгоритму. Результатом данного этапа является построение дерева
* Извлечение из дерева подстановки, являющейся решением.

Следовательно, контекстную диаграмму наиболее целесообразно изобразить в декомпозированном виде.

Декомпозиция контекстной диаграммы представлена на рисунке 3.2.



Рисунок 3.2 – Декомпозиция контекстной диаграммы

Во время лексического и синтаксического анализа производится формирование базы знаний, в которой содержатся уникальные идентификаторы символов, используемых в логическом выводе. Для наглядности лучше использовать диаграмму DFD, чтобы показать на них потоки передаваемых данных.

DFD-диаграмма формирования базы знаний представлена на рисунке 3.3.



Рисунок 3.3 – DFD-диаграмма формирования базы знаний

Дальнейшую декомпозицию целесообразнее представить в виде диаграмм IDEF3 – так как они отражают порядок выполнения действий.

При вводе исходных данных пользователь должен придерживаться лексическим и синтаксическим правилам регламентируемые документацией.

Первым полем для ввода является поле исходных дизъюнктов, которое состоит из множества высказываний разделяемых точкой. Каждое высказывание состоит из предикатов разделяемых «логическим или». При неверном вводе система предупреждает пользователя о местах в которых возможно была совершена ошибка.

Следующим полем для ввода является поле выводимых дизъюнктов. Правила их ввода в точности совпадают с правилами ввода исходных дизънктов.

Данные действия описаны в диаграмме ввода исходных данных, приведенной на рисунке 3.4.



Рисунок 3.4 – Диаграмма ввода данных

Для работы приложения необходима база данных.

Базу данных наглядно можно представить в виде ER-модели данных.

Базовыми понятиями ER-модели данных (ER — Entity-Relationship) являются:

— сущность;

— атрибут;

— связь.

Сущность — это класс однотипных объектов, информация о которых должна быть учтена в модели. Сущность имеет наименование, выраженное существительным в единственном числе, и обозначается в виде прямоугольника с наименованием.

Атрибут сущности — это именованная характеристика, являющаяся некоторым свойством сущности. Наименование атрибута должно быть выражено существительным в единственном числе (возможно, с описательными оборотами или прилагательными). Атрибуты изображаются в прямоугольнике, обозначающем сущность.

Ключ сущности — это не избыточный набор атрибутов, значения которых в совокупности являются уникальными для каждого экземпляра сущности. При удалении любого атрибута из ключа нарушается его уникальность. Ключей у сущности может быть несколько. На диаграмме ключевые атрибуты отображаются подчеркиванием.

Связь — это отношение одной сущности к другой. Каждая связь имеет одно или два наименования. Наименование обычно выражается неопределенной формой глагола. Каждое из наименований относится к своему концу связи. Иногда наименования не пишутся ввиду их очевидности.

Связь типа один-к-одному означает, что один экземпляр первой сущности связан точно с одним экземпляром второй сущности.

Связь типа один-ко-многим означает, что один экземпляр первой сущности связан с несколькими экземплярами второй сущности.

Связь типа многие-ко-многим означает, что каждый экземпляр первой сущности может быть связан с несколькими экземплярами второй сущности, и наоборот. Этот тип связи является временным, допустимым на ранних этапах разработки модели. В дальнейшем такую связь необходимо заменить двумя связями типа один-ко-многим путем создания промежуточной сущности.

Каждая связь может иметь одну из двух модальностей связи:

— модальность «может» (обозначается пунктирной линией);

— модальность «должен» (обозначается прямой линией).

Связь может иметь разную модальность с разных концов.

В ходе логического вывода создается множество копий дизъюнктов и нет необходимости хранить названия в каждом из них. Для этих целей была создана таблица символов, которая хранит имя символа и его тип. В результате применения операций логического вывода формируется множество переменных с одинаковыми именами, но для отображения их принадлежности к определенному шагу вводится индекс, который так же удобно хранить в базе данных.

Схема базы данных представлена на рисунке 3.5.



Рисунок 3.5 – Схема базы знаний

# Объектно ориентированный подход

ООП – это подход, описывающий систему с точки зрения взаимодействия объектов.

При объектно-ориентированном подходе первой обычно строят диаграмму вариантов использования, показывающую совокупность вариантов использования и исполнителей, а также отношения между ними.



Рисунок 4.1 – Диаграмма вариантов использования



Рисунок 4.2 – Диаграмма вариантов использования

Диаграммы взаимодействия моделируют взаимодействия между объектами системы.

На диаграмме взаимодействия отображают один из процессов обработки информации в варианте использования.



Рисунок 4.3 – Диаграмма взаимодействия

На основе диаграммы взаимодействия строится диаграмма кооперации. Кооперативная диаграмма организована вокруг самих объектов. Здесь отображается поток данных. Графически такая диаграмма представляет собой граф из вершин и ребер. Диаграмма кооперации представлена на рисунке 4.4.



Рисунок 4.4 – Диаграмма кооперации

Элементы на диаграмме состояний отображают жизненный цикл одного объекта, начиная с момента его создания и заканчивая разрушением.

Главное назначение диаграммы состояний — описать возможные последовательности состояний и переходов, которые в совокупности характеризуют поведение моделируемой системы в течение всего ее жизненного цикла.



Рисунок 4.5 – Диаграмма состояний

Диаграмма классов — это диаграмма языка UML, на которой представлена совокупность декларативных или статических элементов модели, таких как классы с атрибутами и операциями, а также связывающие их отношения.

С помощью диаграмм классов разработчики могут видеть и планировать структуру системы еще до фактического написания кода, благодаря чему с самого начала могут понять, хорошо ли спроектирована система.

В диаграмме классов описаны классы и методы, используемые системой для проведения операций сервисного обслуживания.

В ходе анализа предметной области были выявлены основные элементы логики предикатов: переменная, константа, функциональная константа и предикат. Функциональная константа, как и предикат, является неким контейнером переменных, констант и функторов. Все эти элементы имеют общую структуру, в результате чего было решено реализовать абстрактный класс «символ», от которого наследуются все остальные элементы. Диаграмма классов Symbol, Constant, FuncConstant и Variable представлена на рисунке 4.5

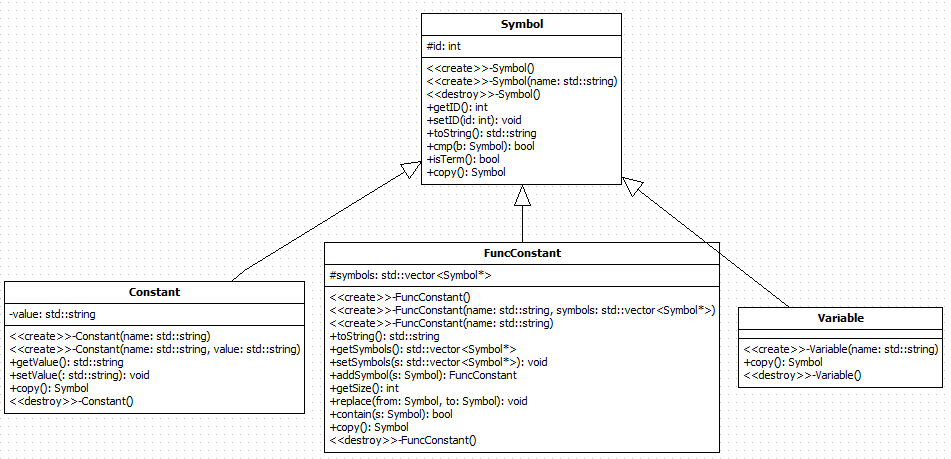


Рисунок 4.6 – Диаграмма классов Symbol, Constant, FuncConstant

Класс Symbol имеет метод cmp, который производит сравнение двух символов на эквивалентность. Если они равны, то возвращается значение истина, иначе ложь.

В классе FuncConst реализованы методы проверки наличия символа в данной функциональной константе (contain) и метод replace, который выполняет подстановку символа.

Базовым элементом дизъюнкта является предикат, который может принимать значение истина или ложь. Предикат может содержать в себе не нулевое количество символов. Из цепочки предикатов строится дизъюнкт. Дизъюнкт в программном коде может быть представлен как массив предикатов.

На рисунке 4.6 представлена диаграмма класса Predicate, в котором соответственно описывается структура предиката.

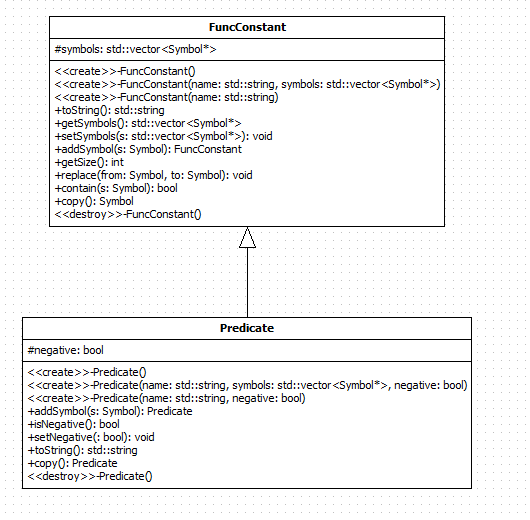


Рисунок 4.7 – Диаграмма классов Statement и Predicate.

Результатом выполнения процедуры унификации является подстановка, которая представляет из себя массив элементов замен. Диаграмма класса Lambda, который хранит элементы структуры подстановки, представлена на рисунке 4.7.

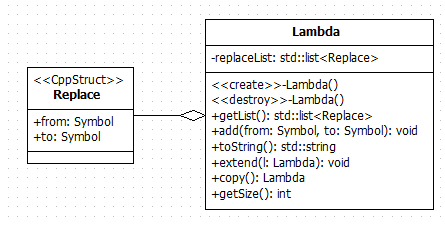


Рисунок 4.8 – Диаграмма классов Lambda и Replace

Множество предикатов образуют высказывания На рисунке 4.8 представлен класс Statement, описывающий структуру высказывания

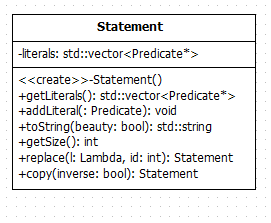


Рисунок 4.9 – Диаграмма класса Statement

Результатом выполнения логического вывода является построение дерева вывода, в вершинах которого находятся промежуточные результаты. Основным классом является класс step, который содержит указатели на других потомков, исходные высказывания, выводимое высказывание, остатки и результаты выполнения операций полного деления, который описан в классе Omega. Диаграмма класса Step представлена на рисунке 4.9

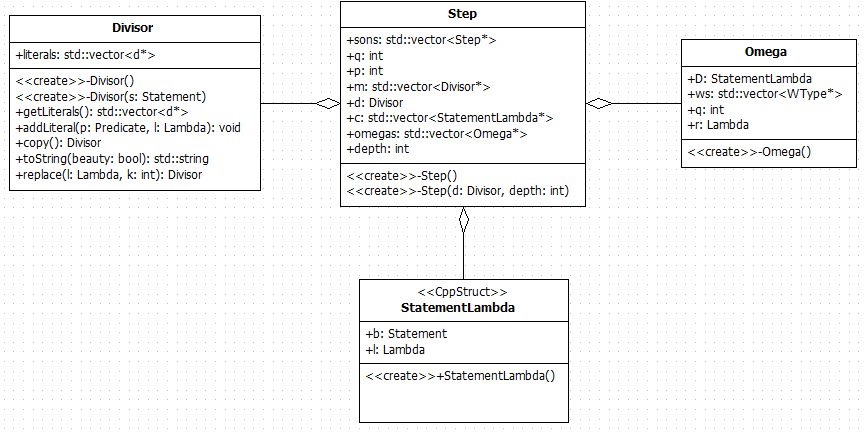


Рисунок 4.10 – Диаграмма класса Step

Класс Omega в свою очередь содержит результаты выполнения полного деления: исходные дизъюнкты, выводимый дизъюнкт, признак q, и указатель на структуру класса, содержащего результаты частичного деления, класс Wtype. Диаграмма класса представлена на рисунке 4.10

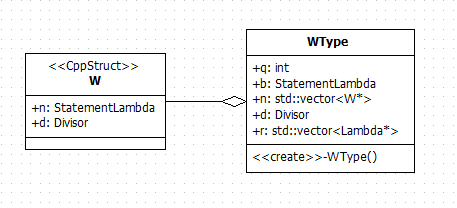


Рисунок 4.11 – Диаграмма класса Wtype

# Вывод

В результате выполнения данного комплекса лабораторных работ изучены структурный и объектно-ориентированный подходы к проектированию информационных систем. Также разработаны различные диаграммы, упрощающие последующее написание программного продукта.

Каждый из рассматриваемых подходов имеет свои преимущества и недостатки. В зависимости от решаемых задач более целесообразно использовать тот или иной подход.

К достоинствам структурного подхода можно отнести:

— возможность проведения глубокого анализа бизнес-процессов, выявления узких мест;

— применение диаграмм IDEF0, IDEF3 и DFD обеспечивает логическую целостность и полноту описания, необходимую для достижения точных и непротиворечивых результатов.

Недостатком структурного подхода может быть ситуация, когда при увеличении количества уровней представления анализ и модификация моделей становятся затруднительными.

К достоинствам объектно-ориентированного подхода можно отнести:

— сравнительная легкость, наглядность, эффективность моделей;

— возможность автоматической генерации кода на основе построенных моделей.

Недостатки объектно-ориентированного подхода:

— невозможность проведения детального анализа бизнес-процессов;

— неполнота и незавершенность некоторых видов диаграмм, возможность их неверной интерпретации.

Таким образом, основным фактором для выбора подхода является решаемая задача. Также немаловажным является то, насколько аналитик знаком со структурным или объектным подходом.

Для решения данной задачи оптимальным является структурный подход.