Интеллектуальные системы и технологии

**Методические указания**

**по выполнению лабораторной работе 1.**

**Оглавление**

[Введение 4](#_Toc88405266)

[Лабораторная работа 1. Разработка экспертной системы. 5](#_Toc88405267)

[Необходимая справочная информация. 7](#_Toc88405268)

[Рекомендуемая литература 14](#_Toc88405269)

# Введение

Выполнение лабораторных работ состоит из двух частей: выполнение практического задания и защита отчета по лабораторной работе. В ходе выполнения лабораторной работы студент должен применить теоретические знания для решения поставленных задач, а также получить практические навыки использования программно-инструментальных средств. В процессе защиты отчета по лабораторной работе, студенту задаются вопросы по теоретической и практической части.

К отчету по лабораторной работе предъявляются следующие требования:

1. к содержанию;
2. к оформлению.

Требования к содержанию отчета указаны в соответствующем разделе для лабораторной работы.

Требования к оформлению отчета определяются соответствующими стандартами (СТО ИрГТУ.005-2009 и СТО ИрГТУ.027-2009).

# Лабораторная работа 1. Разработка экспертной системы.

### Цель работы

1. Закрепление теоретических знаний полученных на лекционных занятиях и в ходе самостоятельной работы.
2. Получение практических навыков формализации знаний и разработки интеллектуальных информационных систем для решения задач в конкретных предметных областях.
3. Освоение программно-инструментальных средств разработки систем основанных на знаниях.

### Требования к выполнению лабораторной работы.

1. В ходе выполнения лабораторной работы студент должен разработать экспертную систему, консультирующую пользователя в решении некоторой проблемы для конкретной предметной области.
2. Знания о предметной области необходимые для решения поставленной задачи должны быть представлены в базе знаний разрабатываемой экспертной системы с использованием продукционной модели.
3. Студент должен сформулировать набор правил как на естественном языке, так и в формальном виде (с использованием синтаксиса средств разработки).
4. По результатам выполнения лабораторной работы оформляется отчет в соответствии с установленными требованиями.

### Содержание задания на выполнение лабораторной работы.

Разработать экспертную систему для решения конкретной задачи в рамках выбранной предметной области. Реализовать подсистему взаимодействия с пользователем в виде диалога в текстовом режиме: пользователь предоставляет системе необходимую информацию, последовательно отвечая на предлагаемые вопросы. В результате диалога экспертная система должна предложить рекомендации по совершению действий, наиболее адекватных полученной информации.

### Варианты индивидуальных заданий.

1. Диагностика неисправностей компьютера
2. Управление поставками в сети супермаркетов
3. Политическая аналитика.
4. Прогнозирование экономической ситуации в стране.
5. Выбор наиболее подходящего автомобиля.
6. Классификация животного по его внешнему виду.
7. Профориентация абитуриента в зависимости от его интересов или способностей.
8. Стратегия вывода фирмы из кризисной ситуации.
9. Прогнозирование ситуации на фондовом рынке.
10. Выбор средств реализации, наиболее подходящих для проекта.

***Примечание****: Данный список содержит примерные варианты предметных областей для выполнения лабораторной работы. Студентам рекомендуется самостоятельно выбрать знакомую им предметную область и сформулировать название темы, согласовав его с преподавателем.*

### Порядок выполнения лабораторной работы.

1. Выбрать предметную область и сформулировать проблему, в решении которой может быть полезна экспертная система.
2. Сформулировать правила на естественном языке, обобщающие знания необходимые для решения поставленной проблемы (не менее 12 правил).
3. Самостоятельно освоить программную среду для разработки экспертных систем CLIPSWindows.
4. Формализовать составленные ранее правила, используя конструкцию defrule.
5. Средствами CLIPSWindows реализовать взаимодействие с пользователем в виде диалога в текстовом режиме (например, в виде функции, позволяющей задавать пользователю произвольный вопрос и предлагающую варианты ответов yes/no).
6. Проверить работоспособность системы, устранить ошибки в коде и противоречия в базе знаний.
7. Составить отчет.

### Требования к содержанию отчета.

1. Введение.
2. Описание предметной области и постановка задачи.
3. Заключительные рекомендации.
4. Словесное описание системы правил.
5. Результаты тестирования.
6. Листинг.
7. Список использованных источников.

***Примечание****: Указанные разделы должны присутствовать в отчете в обязательном порядке; при необходимости допускается дополнять содержание и другими разделами, если этого потребует специфика предложенной студентом темы.*

### Контрольные вопросы.

1. Понятия: знания, данные, информация.
2. Классификация знаний. Отличия знаний и данных. Свойство активности для знаний.
3. Методы извлечения знаний.
4. Понятия базы знаний (БЗ) и модели представления знаний.
5. Продукционная модель представления знаний.
6. Фреймовая модель представления знаний.
7. Понятие экспертной системы и экспертной оболочки.
8. Основные компоненты экспертной системы.
9. Основные элементы интерфейса и синтаксиса среды CLIPSWindows (команды clear, reset, run, конструкторs deffacts, defrule и т.д.).

### Программное обеспечение

В качестве базовых программно-инструментальных средств используется специализированная среда разработки экспертных систем CLIPSWindows.

# Необходимая справочная информация.

## CLIPSWindows

После запуска исполняемого файла открывается диалоговой окно, представляющее собой своего рода командную строку оболочки.

Окно DialogWindow представляет своего рода командную строку, в которой можно выполнять команды CLIPS. Создать новый файл для записи программы можно комбинацией Ctrl-N.

Для того, чтобы выполнить команды, написанные в окне редактирования файла, содержащего программный код, необходимо выделить эти команды и нажать Ctrl-M.

Чтобы загрузить команды из существующего файла используется функция **load**:

*(load “путь к файлу”)*

Команда (facts) выводит список фактов, которые на данный момент находятся в памяти. Команда (rules) выводит список правил.

## Факты

Одной из основных форм представления информации в CLIPS являются факты. Наиболее простым видом фактов являются упорядоченные факты. Каждый такой факт представляет собой список атомарных значений примитивных типов данных, заключенных в скобки, например:

*(this is fact one) (this is fact 2)*

Первое поле должно иметь тип symbol, так же в качестве первого поля не могут быть использованы зарезервированные слова, такие как test, and, or, not, declare, logical, object, exist и forall. В целом же поля могут хранить данные любого примитивного типа CLIPS.

Для добавления фактов в текущий список фактов системы используется команда **assert**, для удаления — команда **retract**.

*(assert <факт> <факт> …) (retract <адрес факта> ...)*

Чтобы просмотреть список уже добавленных фактов в системе можно вызвать команду (**facts**) или выбрать пункт меню *Window* – *Facts Window*

Например, если ввести в диалоговом окне выражение

*(assert (new fact))*

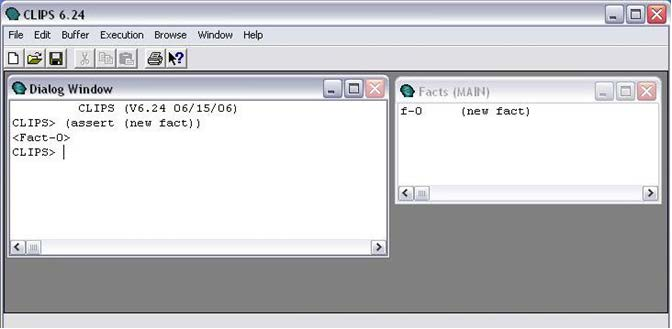
факт (new fact) появится в списке активных фактов системы (рис.1).

Рисунок 1 – Интерфейс CLIPSWindows: рабочее окно и окно фактов.

Более мощным способом представления фактов являются неупорядоченные факты (шаблоны), которые позволяют задать абстрактную структуру факта. Шаблоны определяются конструктором **deftemplate**:

*(deftemplate <Имя шаблона>*

*<Слот1>*

*<Слот2>*

*…)*

Слот может быть как простым, так и составным. В простом слоте хранится одно значение примитивного типа, в составном может храниться список примитивных значений. Например:

*(deftemplate album (multislot title) (slot year)*

*(multislot singer))*

С помощью команды (**reset**) выполняется очистка списка фактов и загрузка в него заранее определенных фактов.

Для указания, какие факты будут автоматически добавлены в систему после сброса среды (reset), используется конструктор deffacts:

*(deffacts <Имя списка>*

*<факт>*

*<факт>*

*…)*

Таких списков может быть несколько и, кроме того, система автоматически добавляет факт (**initial-fact**) при сбросе среды. Этот факт можно использовать как индикатор момента старта логического вывода.

Если нужно выполнить глубокую очистку среды, то для этого используется команда (**clear**). Эта команда очищает все списки, переменные и проч.

## Правила

Для объявления правил используется конструктор **defrule**:

*(defrule <Имя правила> [<Свойства правила>]*

*<предпосылки>*

*=>*

*<следствия>)*

Свойства правила являются необязательным атрибутом. В свойствах можно указать приоритет правила. Предпосылки составляют левую часть правила и являются образцами, которые сопоставляются с имеющимися фактами. В случае совпадения правило срабатывает. Предпосылки связываются неявным оператором **and**. Если в левой части правила не указана ни одна предпосылка, то для активации правила автоматически используется initial-fact.

Следствия (правая часть правила) представляют собой список действий, выполняющихся при активации правила. Правая часть отделяется от левой символом =>.

*(defrule HelloWorld*

*=>*

*(printout t “Hello world” crlf))*

Это правило не имеет предпосылок, поэтому CLIPS автоматически назначит условием активации этого правила initial-fact. Механизм логического вывода запускается командой (**run**). Таком образом, выполним команды (reset), для того чтобы очистить список фактов и добавить в него initial-fact, и (run) для запуска машины вывода.

*(defrule firstRule (activate the firts rule)*

*=>*

*(printout t «the first rule has been activated» crlf))*

Это правило будет активировано, если в системе будет находиться факт (activate the first rule). Если же нам не важно, каковы одно или несколько полей в факте, приводящем к активации правила, то имеет смысл использовать групповые символы. Для того, чтобы указать, что правило может быть активировано как фактом (activate the first rule), так и (activate the second rule) или фактом, содержащим любой другой номер используем групповой символ простого поля - ?. Например:

*(defrule firstRule (activate the ? rule)*

*=>*

*(printout t “the first rule has been activated” crlf))*

Этот символ показывает, что вместо него может быть подставлено одно любое значение. Кроме того, существует групповой символ составного поля -

$?. Вместо него может быть подставлена последовательность значений (в том числе пустая). Например:

*(defrule firstRule (activate $?)*

*=>*

*(printout t “the first rule has been activated” crlf))*

К активации этого правила приведет любой факт первым полем которого является **activate**.

## Переменные

В случае замены поля факта групповым символом значение, которое этот символ принимает, может быть связано с переменной. Для этого нужно записать переменную сразу за групповым символом. Имя переменной обязательно должно начинаться с буквы. Область видимости переменной ограничивается телом правила. Переменные, которым было присвоено значение в левой части правила можно использовать в правой части правила.

В качестве примера используем шаблон album. Укажем следующие автоматически добавляемые факты:

*(deffacts albums*

*(album (title “I Walk The Line”) (year 1964) (singer Johnny Cash)) (album (title “Small Change”) (year 1976) (singer Tom Waits)))*

И добавим правило:

*(defrule releaseYear*

*(album (title $?t) (year ?y) (singer $?s))*

*=>*

*(printout t “Album “ $?t “ by “ $?s “ was released in “ ?y crlf))* Выполнив (reset) и (run) получим следующий результат: *Album (“Small Change”) by (Tom Waits) was released in 1976*

*Album (“I Walk The Line”) by (Johnny Cash) was released in 1964*

## Функции

Функции можно определять с помощью конструктора **deffunction**:

*(deffunction <имя функции>*

*<параметры>*

*<действия>)*

Функция имеет обязательные и необязательные параметры. Обязательные параметры указываются с помощью группового символа простого поля ?, необязательные параметры — с помощью группового символа составного поля $?.

Напишем функцию, принимающую произвольное число аргументов и возвращающую их количество.

*(deffucntion CountParameters ($?p)*

*(printout t “There are “ (length ?p) “ parameters” crlf))*

Данная функция не имеет обязательных параметров. Используя групповой символ составного поля $? мы указываем, что функция может иметь произвольное число необязательных параметров и привязываем эти параметры к переменной p.

Напишем функцию, позволяющую задать пользователю произвольный вопрос и возвращающую его ответ.

*(deffunction ask (?question) (printout t ?question)*

*(bind ?answer (read))*

*?answer)*

Эта функция имеет один обязательный параметр — текст вопроса. С помощью функции bind мы привязываем к переменной answer пользовательский ввод, получаемый функцией read. Функция возвращает значение, полученное во время последнего действия. В данном случае это переменная answer.

Другой полезной функцией является условный оператор if, который имеет следующий синтаксис:

*(if <выражение> then <действия> [else <действия>])*

Напишем правило, использующее предыдущую функцию для задания вопроса и условный оператор для добавления различных фактов в зависимости от ответа пользователя:

*(defrule exampleRule*

*=>*

*(bind ?response (ask «Do you want to start? Yes/no »)) (if (eq ?response yes)*

*then (assert (start))*

*else (assert (do not start))))*

Как видно, это правило не имеет предпосылок, а значит активирующим фактом для него является системный initial-fact. Функция bind привязывает ответ пользователя к переменной response. В условном операторе используется функция eq, сравнивающая значение переменной response с константой yes. В зависимости от содержимого переменной вводится либо факт (start), либо (do not start).

В этом правиле мы использовали функцию ask, написанную нами ранее. Для корректной работы эта функция должна быть написана и интерпретирована до этого правила.

## Запуск машины вывода (пример «Hello world!»)

Перед запуском процесса логического вывода необходимо выполнить команду (reset). Она осуществляет сброс среды и вводит стартовые факты в список активных фактов.

Для того, чтобы определить, какие факты в системе будут стартовыми используется конструкция deffacts:

*(deffacts <имя списка фактов>*

*<факт>*

*…*

*<факт>)*

Если будет объявлено несколько списков, то все они будут внесены в список стартовых фактов. Например:

*(deffacts list1*

*(fact one) (fact two))*

*(deffacts list2*

*(another fact) (one more fact))*

В результате выполнения команды reset в память будут добавлены указанные четыре факта и еще один факт (initial-fact), который может использоваться для определения момента запуска механизма вывода.

Добавим в систему следующее правило:

*(defrule HelloRule*

*=>*

*(printout t “Hello World!” crlf))*

Как видно в этом правиле отсутствует предпосылка, это означает, что правило будет активировано служебным фактом (initial-fact) при старте процесса вывода.

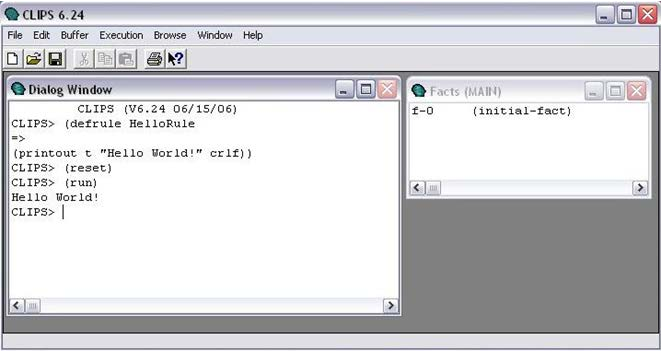
Если выполнить сброс системы (reset) и запустить процесс вывода командой (run) будет выведена строка «Hello World!» (рис.2).

Рисунок 2 – Результаты работы машины вывода.

# Рекомендуемая литература

1. СТО ИрГТУ 005-2009 СИСТЕМА МЕНЕДЖМЕНТА КАЧЕСТВА. Учебно-методическая деятельность. Оформление курсовых и дипломных проектов(работ) технических специальностей. [электронный ресурс], точка доступа: [http://www.istu.edu/docs/sto-005-](http://www.istu.edu/docs/sto-005-2009.doc) [2009.doc](http://www.istu.edu/docs/sto-005-2009.doc) (дата обращения: 01.09.13).

## Частиков, А.П. Разработка экспертных систем. Среда CLIPS / А. П. Частиков, Т. А. Гаврилова, Д. Л. Белов. – СПб.: БХВ-Петербург, 2003. – 606 с. : a-ил. – (Учебное пособие).

1. Методы и технологии искусственного интеллекта / Л. Рутковский; пер. с польск. И. Д. Рудинского. – Москва: Горячая линия -Телеком, 2010. – 519 с.
2. Джексон, П. Введение в экспертные системы: [Пер. с англ.] / Питер Джексон. – 3-е изд. – М.: Вильямс, 2001. – 622 с. : a-ил.
3. Сидоркина, И. Г. Системы искусственного интеллекта: учеб. пособие для студентов высш. учеб. заведений , обучающихся по направлению 230100 «Информатика и вычислительная техника» / И. Г. Сидоркина . – Москва: КНОРУС, 2011. – 245 с.
4. CLIPS User’s guide [электронный ресурс]. Точка доступа: <http://clipsrules.sourceforge.net/documentation/v624/ug.htm>

# 