МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ
«БРЕСТСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

КАФЕДРА ИНТЕЛЛЕКТУАЛЬНЫХ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ПО ДИСЦИПЛИНЕ

ТЕХНОЛОГИЯ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА РОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ»

асть 2. Реализация и сопровождение экспертных систем

для студентов специальности «Искусственный интеллект»

БРЕСТ – 2009

УДК 681.3

В методических указаниях представлены материалы по лекционному курсу и лабораторному практикуму по дисциплине «Технология и инструментальные средства проектирования интеллектуальных систем». Приведены общие сведения о возможностях экспертной оболочки GURU: основные конструкции языка оболочки, организация логического вывода, возможности подсистемы объяснений вывода, действия пользователя в командном и диалоговом режимах взаимодействия с оболочкой. Рассмотрены вопросы реализации и сопровождения экспертных систем: программирование, компилирование, отладка и тестирование, разработка документации и испытание.

В приложениях методических указаний определена процедура организации и проведения цикла лабораторных работ по тематикам «Изучение возможностей оболочки GURU» и «Проектирование, реализация, документирование и испытание прототипа учебной экспертной системы». Для каждой работы приводятся постановка задач, методика, порядок и результат их выполнения.

Данные методические указания рекомендуются студентам по специальности «Искусственный интеллект» при изучении лекционного материала, выполнении лабораторных и самостоятельных работ по дисциплине «Технология и инструментальные средства проектирования интеллектуальных систем». Пособие рекомендовано студентам очной и заочной форм обучения.

Издается в 2-х частях. Часть 2.

Ил. 10, табл.4, список лит.21 назв.

Составители: В.И. Хвещук, доцент, к.т.н.
Г.Л.Муравьев, доцент, к.т.н.

Рецензент: доцент, к.п.н. Брестского государственного университета им. А.С. Пушкина
А.А. Козинский

СОДЕРЖАНИЕ

1. ЭКСПЕРТНАЯ ОБОЛОЧКА GURU
1.1. ОБЩИЕ СВЕДЕНИЯ ОБ ОБОЛОЧКЕ

Возможности GURU. Экспертная оболочка GURU – это среда для автоматизации разработки экспертных систем (ЭС) продукционного типа. Эта среда обеспечивает следующие возможности:

- автоматизацию процесса создания ЭС и консультацию с ними;
- управление данными;
- формирование специальных запросов и графические средства;
- управление экраном, печатными формами;
- анализ электронных таблиц (ЭТ), генерацию статистики, итоговых отчетов и другие.

Взаимодействие пользователей с GURU. Оболочка GURU обеспечивает пользователям диалоговый режим взаимодействия. Пользователи делятся на разработчиков ЭС и прикладных пользователей, применяемых ЭС для решения задач из конкретных предметных областей (Про). Можно выделить следующие виды диалога:

- **система меню (диалоговый)** – ориентирована на разработчиков ЭС и реализована с помощью развитой системы меню;
- **командный** – обеспечивает выполнение определенного набора команд языка GURU и предназначен для разработчиков ЭС;
- **естественный язык** – диалог, ориентированный на конкретную ПрО и обеспечивается ЭС. Этот вид диалога предназначен для прикладных пользователей. Диалог реализуется на основе запросов ЭС, которые обеспечивают обмен фразами на естественном языке и выдают результаты. Например, система спрашивает "Ваш запрос?". Написав в командной строке фразу "Найти всех работающих 1967 года рождения", пользователь получает от системы разумный ответ.

Обычно применяются смешанные режимы. Находясь в любом месте системы меню, можно получить подсказку по действиям, допустимым в этом меню. Для этого вызывается помощь одновременным нажатием <Ctrl-L>.

Структура экспертной системы. Основными компонентами экспертных систем являются:

- пользовательский интерфейс (ПИ);
- машина логического вывода (МЛВ);
- база знаний ЭС или хранимые экспертизы (программа ЭС на языке оболочки).

Интерфейс пользователя описывает отношения между пользователем и ЭС. Пользователь ставит задачу, а МЛВ должна ее выполнить или объяснить, почему нельзя ее выполнить.

База знаний (хранимые экспертизы) – это набор правил, отображающих знания в конкретной предметной области (ПрО). В основе построения БЗ используется продукционная модель представления знаний или в виде совокупности правил. В каждом правиле есть посылка (IF) и заключение (THEN).

Машина логического вывода (МВ) – это часть программного обеспечения (ПО) оболочки, которое обеспечивает решение задач путей аргументации или вывод заключения на основе анализа базы знаний. Если машина логических выводов признает посылку верной, то и заключение будет верным. МВ обеспечивает прямой и обратный способы вывода для продукционных моделей знаний.

Программа экспертной системы. ЭС в рамках оболочки представляется в виде программы на языке GURU. Программа ЭС имеет фиксированную структуру и состоит из следующих разделов: цели, инициализации переменных, завершающих действий, правил вывода и определения переменных. Для запуска ЭС ее программу необходимо транслировать, а затем запустить на выполнение (режим консультирования или решение задач ЭС).

1.2. ОСНОВНЫЕ КОНСТРУКЦИИ ЯЗЫКА ОБОЛОЧКИ

Основными объектами, из которых конструируется текст программы ЭС в рамках оболочки GURU, являются:

- ***Константы;***
- ***Переменные;***
- ***Выражения;***
- ***Функции;***
- ***Операторы (команды);***
- ***Правила;***
- ***Другие объекты оболочки*** GURU (макроопределения, шаблоны и др.).

1.2.1. КОНСТАНТЫ

Константа – это фактическое значение данных, которое система может использовать во время работы. Имеется 4 типа констант:

- ***строки*** (произвольные последовательности символов, заключаемые в двойные кавычки);
- ***числа***;
- ***логические константы*** (***TRUE, FALSE***);
- ***неизвестная константа*** (***UNKNOWN***).

Использование констант в системе GURU практически не отличается от их использования в обычных языках программирования.

1.2.2. ПЕРЕМЕННЫЕ

Переменная – объект системы, значение которого может меняться в процессе обработки. Существует 5 типов переменных:

- ***Символьные;***
- ***Числовые;***
- ***Целочисленные;***
- ***Логические;***
- ***Неизвестные.***

Кроме того, имеется 4 класса переменных:

- ***поля*** (применяемые в таблицах записей);
- ***ячейки*** (используемые при обработке электронных таблиц);
- ***предварительно определенные*** в оболочке;
- ***рабочие.***

Наиболее широко используемыми из этих классов являются рабочие и предварительно определенные переменные.

Рабочие переменные – это переменные общего назначения, используемые для хранения результатов вычислений, операций ввода-вывода, выполнения правил и т.д. Эти переменные, как и константы, во многом аналогичны переменным, применяемым в обычных языках программирования. Первоначально все рабочие переменные имеют значения ***UNKNOWN.*** Им можно присвоить значение любого типа. Простейший способ задания значения переменным – с помощью операторов присваивания:

[LET] < имя_переменной> =< значение>

или с помощью команды ввода:

INPUT <имя_переменной> [тип] [WITH <строка_подсказки>]

Здесь части, заключенные в квадратные скобки, необязательны, а выражения в угловых скобках заменяются на конкретные имена переменных или их значения. Указание типа переменной при ее вводе обязательно только для числовых и целочисленных переменных (типы ***NUM*** и ***INT*** соответственно). По умолчанию переменная при вводе считается символьной.

Примеры операторов присваивания:

COST = 5000; MONITOR = "ЦВЕТНОЙ"; RISK = TRUE

Пример команды ввода:

INPUT SCORE WITH "ВВЕДИТЕ СРЕДНИЙ БАЛЛ "

Для вывода данных используется команда: **OUTPUT <выходные данные>**. Например,

OUTPUT "РЕКОМЕНДАЦИЯ: ", RECOM

Предварительно определенные переменные – это переменные, определяемые самой системой. Они делятся на две группы: переменные среды и утилитные переменные.

Переменные среды – это 112 переменных, описывающих среду оболочки GURU. Они предназначены для управления обработкой факторов уверенности, логическим выводом, вводом-выводом данных и т.д. Значения переменных среды могут быть изменены пользователем с помощью операторов присваивания, так же как и для обычных (рабочих) переменных. Список основных переменных среды приведен в приложении 1. Например, изменение значения переменной среды (установка максимальной длины строки при выводе на экран равной 30) можно определить в разделе инициализации следующим образом: *E.LSTR = 30*

Утилитные переменные – это переменные, значения которых изменяет сама оболочка. Они используются, главным образом, для отслеживания процесса и результатов обработки наборов правил.

1.2.3. НЕЧЕТКИЕ И НЕДОСТОВЕРНЫЕ ЗНАНИЯ

В рамках оболочки возможно использование ***нечетких переменных***, т. е. имеющих одновременно несколько значений. При этом каждому такому значению может быть присвоен ***фактор уверенности*** – число в диапазоне от 0 до 100, определяющее степень уверенности пользователя в данном значении переменной. Представление нечетких и недостоверных знаний, а также организация работы с ними приведена в п.1.5.

1.2.4. ОПЕРАЦИИ И ФУНКЦИИ

В рамках оболочки GURU определены следующие операции над данными: арифметические; сравнения; логические; строковые. В выражениях можно использовать следующие функции: числовые, символьные, логические. Перечень основных операций и функций для применения в выражениях приведен в приложении 2.

1.2.5. ВЫРАЖЕНИЯ

Выражения с переменными в оболочке GURU могут состоять из одной или нескольких переменных и (или) констант. В зависимости от типа участвующих в выражении операндов результат может представлять собой строку, число, логическое или неизвестное значение. В выражениях могут использоваться обычные арифметические операции, а также функции. Пример выражений с числовыми значениями переменных:

2 + 4 DEPTH=5 6 + DEPTH 22 SQRT (4) 60/5 5.67 * PI**

Пример выражений над строчными переменными:

NAME = "Иванов" + "Иван" – сцепление NAME становится равным "Иванов Иван".

Пример выражений с логическими операциями:

A = B A<>B A <= B A >= B

Для обозначения равносильности одного элемента другому применяется оператор **IN** (проверки того, соответствуют ли друг другу правая и левая части выражения). Допустим, необходимо найти служащего, чья фамилия Минев или Манев или Монев, тогда вводим логическое выражение:

NAME IN [M$HEB].

Пример использования составных логических выражений:

15 > 9 AND 20 < 100 – истинное выражение;

9 > 15 AND 20 < 100 – ложное выражение.

1.2.6. ПРАВИЛА

База знаний ЭС в оболочке GURU базируется на правилах. Отдельное правило состоит из посылки (IF) и заключения (THEN). **Посылка правила** может включать:

- различные типы и виды переменных, поддерживаемых GURU;
- логические операторы (EQ, NE, GT, GE, LT, LE, IN, AND, OR, XOR, NOT);
- числовые операторы (+, -, /, *, **);
- числовые функции (SIN, COS и т.д.);
- символьные функции.

Заключение правила может включать команды:

- присвоение значения различным переменным;
- организации консультирования с другим набором правил;
- другие команды GURU и т.д.

Правила хранятся в обычном текстовом файле. Пример:

```
RULESET:    EASYCALC
GOAL:       INTRATE
RULE:       R1
    IF: MONTHPAY < 50
    THEN: PERIOD = 120
RULE:       R2
    IF: PERIOD > 90
    THEN: INTRATE =12.5
```

Здесь EASYCALC – имя набора правил (RULESET указывать не обязательно);
INTRATE – имя переменной цели;
R1, R2 – имена правил;
PERIOD, INTRATE, MONTHPAY – переменные.

1.2.7. КОМАНДЫ

Все команды (операторы) языка оболочки GURU можно разделить на группы:

1. Команды, которые используются только в тексте программы ЭС. Они приведены при рассмотрении разделов программы в п.1.6.2 – п.1.6.6.
2. Команды, которые используются только в командном режиме взаимодействия с оболочкой GURU. Перечень команд приведен в приложении 5.
3. Общие команды, применяемые как в тексте программы ЭС, так и в командном режиме. К ним относятся команды: присвоения значений переменным, ввода, вывода, обработки электронных таблиц, обеспечения графических возможностей.

1.2.8. СТРУКТУРА ПРОГРАММЫ

Программа ЭС в рамках оболочки GURU представляет собой текстовый файл с расширением .RSS, в котором содержится исходный текст программы на входном языке оболочки. Для выполнения ЭС или консультирования ЭС (режим решения задач ЭС) необходимо компилировать текст программы, а затем запустить его на выполнение.

Процесс написания текста программы ЭС реализуется либо в диалоговом режиме под управлением оболочки, либо выполняется в любом текстовом редакторе с выполнением требований синтаксиса конструкций входного языка оболочки, а затем этот текстовый файл используется аналогично тексту программы, подготовленному в оболочке.

Структура исходного текста программы ЭС имеет фиксированную конструкцию и состоит из следующей совокупности разделов:

- ***раздел определения цели ЭС***

- ***раздел инициализации переменных ЭС***
- ***раздел завершающих действий ЭС***
- ***раздел правил вывода ЭС***
- ***раздел определения переменных ЭС***

конец программы ЭС

В соответствии с приведенной структурой программа состоит из пяти разделов, каждый из которых состоит из определенной совокупности операторов (команд). Каждый отдельный раздел имеет фиксированное начало, которое определяется соответствующим ключевым словом (см. рис.1.1). Конец программы фиксируется ключевым словом ***END***.

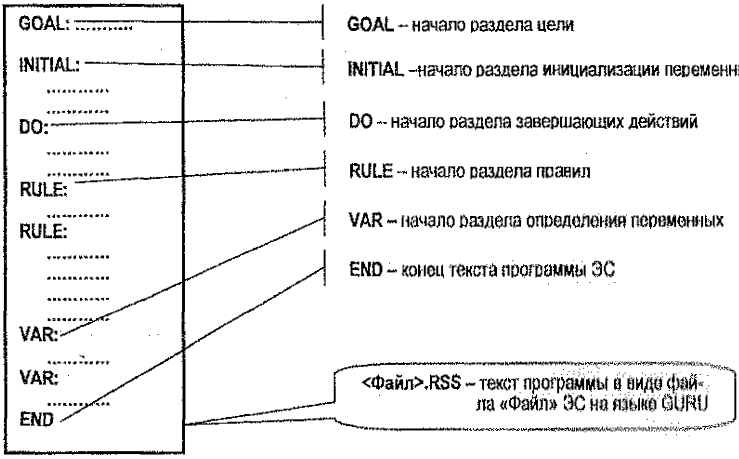


Рисунок 1.1 – Структура программы ЭС на языке оболочки GURU

Пример текста программы ЭС на входном языке оболочки приведен на рис.1.9. и рассмотрен в п.1.10.

1.2.9. КОМАНДЫ ОБРАБОТКИ ЭЛЕКТРОННЫХ ТАБЛИЦ

При выполнении расчетов оказывается удобным исходные данные и результаты расчетов представить в виде таблицы. Для автоматизации табличных расчетов используются представления данных в виде электронных таблиц (ЭТ).

Электронная таблица GURU – это таблица, которая упрощает процесс работы с числами. ЭТ – это матрица величиной 255 на 255 элементов, в которой цифры идентифицируют строки, а буквы – столбцы.

Ячейка – это пересечение строки и столбца. Каждая ячейка обозначается знаком #, за которым следует указание местоположения столбца или строки (например, # A1 определяет ячейку в верхнем левом углу ЭТ).

Для того, чтобы войти в режим обработки ЭТ, необходимо ввести в командном режиме команду CALC. Синтаксис этой команды:

CALC < размерность >, где < размерность > (ЭТ) – два выражения, отделенные запятыми, указывающие число строк и число столбцов.

При входе в режим обработки ЭТ можно давать все команды, определенные в GURU. При этом перед каждой командой необходимо давать символ "\". Например: \CLEAR \PERFORM MY_PROGRAM \OUTPUT "Это команда OUTPUT"

В GURU есть дополнительные команды, определенные только для ЭТ.

Если размерность ЭТ отсутствует, то по умолчанию ЭТ определяется размером 30 строк на 30 столбцов. Например, CALC 10,20 – определяет ЭТ размером 10 строк и 20 столбцов.

После ввода команды CALC на экран выводятся границы и ячейки ЭТ, область состояния и строка ввода. Область состояния состоит из следующих строк:

- вспомогательной строки (первая строка) – используется для более совершенной обработки ЭТ;
- строки ввода (вторая строка). Используется для указания дальнейших действий и ввода команд;
- строки вывода (третья строка) – выводит определение (если оно существует) текущей ячейки, а также выводит сообщения об ошибках.

Если в строке ввода ввести \< команда >, то выполняется команда обработки ЭТ.

Если же косая черта (\) отсутствует, то GURU считает, что вводится определение ячейки.

Определение ячейки означает, что ячейке можно присвоить значение константы или какую-нибудь функцию. Например, ячейка может иметь определение sin (#A2) – это означает, что ячейка имеет значение синуса содержимого ячейки #A2.

Когда задается команда расчета ЭТ, то она сразу выполняется.

Если в командной строке вводится \< команда >, то GURU выводит сообщение: "*ЗАНЯТО*". Когда GURU заканчивает обработку команды, то это сообщение исчезает.

Клавишами управления курсором можно перемещаться от одной ячейки к другой. Текущая ячейка будет выведена на экране.

Ячейки ЭТ могут выступать в качестве переменных в программе. Имя ячейки должно начинаться с символа "#". Ячейке ЭТ можно присвоить какое-нибудь значение.
Например: #C2=18.

Перечень команд и их формат для обеспечения обработки ЭТ приведен в приложении 3. Для организации обработки ЭТ используются следующие команды:

\EDIT – вывод определения текущей ячейки, которая в дальнейшем может быть отредактирована или изменена с помощью клавиш управления курсором. Установите курсор на ячейке, которую необходимо редактировать. После команды \EDIT можно изменить определение этой ячейки.
\UNDEFINE – удалить существующее определение ячейки, включающее значение, тип представления и шаблон ячейки.
\USING – задать шаблону ячейки или блоку ячеек.
\WIDTH – задать ширину столбцов. По умолчанию ширина столбцов равна 9.
\COMPUTE – вычислить все значения ячеек ЭВ, основанные на текущих определениях.
\LET – присвоить переменной значение.
\COPY – копировать определение одной ячейки в другую.
\SAVE – сохранить ЭТ в файле.
\LOAD – загрузить ЭТ из файла.
\STOP – закрыть ЭТ и вернуться в командный режим "GURU".
\DISPLAY – вывод на печать необходимых участков или всех значений ячеек ЭВ.
\DUMP – печать всей информации относительно заданных ячеек, которая включает: имя ячейки, текущее значение, определение (если существует) и шаблон (если существует).

1.2.10. ГРАФИЧЕСКИЕ СРЕДСТВА GURU

Управление графиками с помощью утилитных переменных и переменных среды. С помощью команд графики можно создать различные графы на основе данных из ЭТ, статистики рабочих переменных и массивов. Графы системы GURU могут создаваться на заказ с помощью цветов, моделей, диапазонов масштабов. Они могут сохраняться, печататься и вычерчиваться. Когда необходимо вычерчивать граф, то по умол-

чанию он занимает весь экран. При необходимости граф может занимать указанкую часть экрана. При этом если не указан диапазон выводимых данных, он рассчитывается автоматически так, чтобы занимать максимальный экран.

Некоторые из переменных типа "среда" и "утилита" предназначены для построения и оформления графиков. Основные переменные следующие:

- **E.BACG** определяет фоновый цвет графического экрана;
- **E.DECI** определяет число цифр справа от десятичной точки и числах оси и процентах круговой диаграммы;
- **E.FOHG** определяет основной цвет графического экрана;
- **#TITLE** используется как заголовок графа.

Кроме этих, имеются определенные переменные, которые используются только для графики и приведены в таблице 1.1.

Таблица 1.1 – Утилитные переменные и переменные среды для управления графиками

Имя	Значение	Тип	Значение по умолчанию
E.GRID	Фоновая сетка для графов	Логический	TRUE
E.WFU	Перед удалением графического экрана с дисплея подождать, пока пользователь не нажмет на клавишу	Логический	TRUE
#XLABEL	Метка, которая должна использоваться на оси X графа	Символьный	
#YLABEL	Метка, которая должна использоваться на оси Y графа	Символьный	

Переменная E.GRID управляет воспроизведением фоновых сеток для графов. Сетка состоит из горизонтальных и вертикальных линий. Если E.GRID = TRUE, то сетка появится. Приведем перечень команд для работы с графическими средствами GURU:

PLOT BAR – черчение гистограммы по исходным данным.

PLOT PIE – черчение круговой диаграммы ("пирог"). Приведем один из вариантов написания команды: **PLOT LABELED PERCENT PIE FROM**. Команда должна сопровождаться параметрами: <источник данных> AT <размещение> <источник данных> – численное выражение, блок ячеек или блок массивов ЭТ. Если "FROM <источник данных>" опущено, то используется <источник данных> предыдущей команды PLOT.

Слово PERCENT в команде – необязательное. Если оно присутствует, то рядом с сегментами круговой диаграммы воспроизводятся проценты.

Слово LABELED может использоваться только, если все блоки в источнике данных имеют точно две колонки. Когда мы строим круговую диаграмму с использованием LABELED, значения отсчетов во второй колонке каждого блока используются для указания размеров сегментов. Эти сегменты маркированы соответствующими значениями из первых колонок (до 6 символов в каждом обозначении). AT < размещение> используется аналогично команде PLOT BAR. Примеры:

PLOT LINE – чертить линейный граф.

PLOT FUNCTION – чертить функциональную линию.

RANGE – задать диапазон значений осей координат, используемых при построении графиков и контролирует расстояние между отметками на осях.

PATTERN – задать исходные данные – тип заполнения, строку, отметку и типы цветов, используемых при вычерчивании графа.

PLOT TO – сохранить графический экран в файле. PLOT TO < имя файла >

PLOT FROM – загрузить графический экран с диска. PLOT FROM < имя файла >

Перечень команд и их формат для обеспечения графических возможностей оболочки приведен в приложении 4.

1.2.11. ПРЕДСТАВЛЕНИЕ НЕЧЕТКИХ И НЕДОСТОВЕРНЫХ ЗНАНИЙ
1.2.11.1. ФАКТОРЫ УВЕРЕННОСТИ

Задачи, решаемые с использованием ЭС, обычно связаны с неопределенностью. Она может быть вызвана тем, что некоторые факторы, влияющие на решение задачи, могут быть известны недостоверно. Кроме того, задачи, решаемые в ЭС, имеют, как правило, много возможных вариантов решений (как промежуточных выводов, так и окончательных заключений), каждое из которых может быть выбрано с определенной степенью уверенности в зависимости от различных факторов. В связи с этим для большинства ЭС оказывается необходимым наличие механизма представления и обработки различных степеней достоверности экспертных знаний.

Основными средствами для учета неопределенности и недостоверности знаний в ЭС являются факторы уверенности и нечеткие (многозначные) переменные.

Факторы уверенности – это числа (обычно в диапазоне от 0 до 1 иди от 0 до 100), отражающие степень уверенности в некотором элементе знаний, используемых в ЭС (таким элементом может быть значение переменной, продукционное правило).

Фактор уверенности в системе GURU – это целое число от 1 до 100, связываемое со значением переменной. Эта величина отражает степень уверенности в том, что переменная имеет именно данное значение. Если фактор уверенности не указан, он считается равным 100 (полная уверенность в том, что переменная имеет именно данное значение).

Фактор уверенности отражает степень достоверности знаний и в общем случае не представляет собой значения вероятности, хотя вероятность и может использоваться в качестве значения фактора уверенности. Так как фактор уверенности это не вероятность, сумма факторов уверенности для всех значений переменной (о переменных, имеющих несколько значений, м. ниже) не обязана быть равной 100.

Фактор уверенности может назначаться переменной в операторе присваивания:

<переменная:> = <значение> CF <фактор уверенности>

Например, переменной X присваивается значение 10 факторам уверенности 80:

```
X = 10 CF 80
```

Значение фактора уверенности может указываться также при вводе значения переменной. Чтобы система вместе со значением переменной запрашивала ее фактор уверенности, необходимо установить значение переменной среды с именем E.ICF:

```
E. ICF = TRUE
```

По умолчанию эта переменная имеет значение FALSE, т.е. фактор уверенности не запрашивается. Выводом фактора уверенности управляет переменная E.OCF; чтобы такой вывод выполнялся, ей заранее необходимо присвоить назначение TRUE.

Значения переменных E.ICF и E.OCF могут изменяться в наборе правил многократно, так как ЭС может использовать как нечеткие, так и однозначные переменные.

Фактор уверенности используется также для того, чтобы устанавливать, в каком случае система должна считать переменную известной.

Переменная считается известной, если ее фактор уверенности превышает пороговое значение, устанавливаемое переменной среды E.UNKN. По умолчанию E.UNKN = 20. Если фактор уверенности переменной не превышает пороговый, то система рассматривает переменную как неизвестную и продолжает поиск правил, в которых можно определить ее значение.

Как правило, факторы уверенности связываются с различными значениями нечетких переменных.

1.2.11.2. НЕЧЕТКИЕ ПЕРЕМЕННЫЕ

Нечеткая переменная – это переменная, которая может иметь одновременно несколько значений, каждое из них – со своим фактором уверенности. Нечеткие переменные предназначены для представления недостоверных знаний, когда точно неизвестно, какое именно значение (из нескольких возможных) имеет переменная. Значения нечеткой переменной перечисляются в фигурных скобках вместе с их факторами уверенности.

Пусть, например, эксперт считает, что спрос на некоторую продукцию в следующем году может составить 1 млн. единиц (но эксперт уверен в этом не более чем на 20%), 2млн. (это, по мнению эксперта, наиболее вероятно на 70%) или 3 млн. (маловероятно – не более 10%). Это можно отразить в виде нечеткой переменной следующим образом:

```
X = {1 CF 20, 2 CF 70, 3 CF 10}
```

Если все значения переменной имеют одинаковые факторы уверенности, это представляется следующим образом:

```
<переменная> = <список значений> CF <фактор уверенности>
```

Пусть, например, переменная X может иметь значения 0 или 1 с факторами уверенности 50:

```
X = {0, 1} CF 50
```

Различные значения нечеткой переменной (со своими факторами уверенности) могут указываться и при ее вводе. Количество запрашиваемых значений устанавливается переменной среды **E.IFUZ**, а количество выводимых (в команде **OUTPUT**) значений – переменной **E.OFUZ**. По умолчанию эти переменные равны 1, поэтому, если какая-либо рабочая переменная должна вводиться или выводиться как нечеткая, значения этих переменных среды должны быть указаны явно.

Максимальное количество значений, возможных для нечеткой переменной, устанавливается переменной среды **E.NUMV** (для всех переменных данного набора правил) или параметром **Limit** (для конкретной переменной). По умолчанию **E.NUMV=4**. В любом случае количество таких значений не может превышать 255.

В набор значений нечеткой переменной можно добавлять новые значения с сохранением всех уже имеющихся; можно также удалять имеющиеся значения из набора. Пусть, например, к набору значений нечеткой переменной X требуется добавить новое значение 3 (с фактором уверенности 20), а затем удалить значение 0 (с фактором уверенности 50):

```
X+=3 CF 20 X-=0 CF 50
```

Если вместо первого из этих операторов использовать следующий: **X = 3 CF 20**, то переменная X получит значение 3 с фактором уверенности 20, но все ее прежние значения будут потеряны.

Нечеткие переменные могут использоваться в посылках и заключениях правил, в различных операциях (арифметических, логических, строковых и т.д.) так же, как и обычные переменные. Результатами таких операций также являются нечеткие переменные.

В некоторых случаях наличие нечетких переменных может привести к излишнему усложнению задачи. Например, при умножении двух нечетких переменных, одна из которых имеет 3 значения, а другая – 4, получается нечеткая переменная с 12 значениями. Количество используемых значений нечеткой переменной можно ограничить с помощью переменной среды **E.BEST**. Она устанавливает количество наилучших (т.е. имеющих максимальный фактор уверенности) значений нечеткой переменной, используемых в процессе обработки набора правил. По умолчанию **E.BEST=4**.

1.2.11.3. ФУНКЦИИ ДЛЯ РАБОТЫ С ФАКТОРАМИ УВЕРЕННОСТИ

Во время консультации с набором правил может возникнуть необходимость в знании определенных фактов о значениях переменной, например, сколько значений имеет переменная, какое значение имеет наивысший или низший фактор уверенности для данной переменной, а также каковы факторы уверенности этих значений. Эту информацию можно получить с помощью приводимых ниже функций.

Таблица 1.2 – Функции для работы с факторами уверенности

CFV (переменная, значение)	Выдает фактор уверенности (CF) данного значения переменной. Если значение не существует, выдается 0.
CFN (переменная, номер)	Выдает CF значения переменной, указанного номером. Порядок следующий: 1 -наивысший CF,2 -второй по величине и т.д.
VALN (переменная, номер)	Выдает значение переменной по номеру.
NUMVAL (переменная)	Выдает количество значений переменной.
HIVAL (переменная)	Выдает значение переменной с наивысшим CF.
HICF (переменная)	Выдает максимальный CF переменной.
LOVAL (переменная)	Выдает значение переменной с наименьшим CF
LOCF (переменная)	Выдает минимальный CF переменной

1.2.11.4. ОБЪЕДИНЕНИЕ ФАКТОРОВ УВЕРЕННОСТИ

Объединение факторов уверенности, т.е. вычисление обобщенного фактора на основе факторов уверенности переменных, участвующих в выражении, требуется в следующих случаях:

1) Объединение факторов уверенности нескольких переменных в посылке правила для определения фактора уверенности этой посылки. Для управления таким объединением используется переменная среды **E.GFJO** (если условия в правиле связаны логической операцией И) или **E.CFCO** (если используется операция ИЛИ).
2) Объединение факторов уверенности нескольких переменных в операции (например, арифметической) для определения фактора уверенности ее результата. Для этого используется переменная **E.CFJO.**
3) Объединение факторов уверенности посылки и заключения для определения общего фактора уверенности заключения таким объединением управляет переменная **E.CFVA.**
4) Объединение факторов уверенности нескольких одинаковых значений нечеткой переменной, полученных в результате применения разных правил. Таким объединением также управляет временная переменная **E.CFVA.**

Кроме того, способ объединения факторов уверенности для отдельных переменных может устанавливаться элементом описания переменных **CF Type**; в этом случае его значение используется для данной переменной вместо значения **E.CFVA.**

Допустимые значения переменных **E.CFJO** и **E.CFJO** и соответствующие им правила объединения факторов уверенности приведены ниже.

Таблица 1.3 – Правила объединения факторов уверенности

Значение	**Правило объединения**	
	E.CFJO	**E.CFJO**
M	CFm = min (CF1, CF2)	CFm = max (CF1, CF2)
P	CFp = CF1*CF2\100	CFp=CF1+CF2-CF1*CF2\100
A	Cfa= (CFm+CFp)\2	Cfa= (CFm+CFp)\2

Здесь CF1 и CF2 – объединяемые факторы уверенности.

По умолчанию используется **E.CFJO = "M", E.CFCO = "M"**. Пусть, например, имеются следующие переменные:

X=10 CF 80, Y=5 CF 40.

Тогда факторы уверенности для посылок **(X>0) AND (Y>0)** и для **(X>0) OR (Y>0)** будут следующими:

Таблица 1.4 – Пример объединения факторов уверенности

Операция в посылке	**Значения**		
	M	**P**	**A**
AND	40	32	36
OR	80	88	84

Как видно, значение фактора уверенности для посылки, условия в которой соединены связкой И, не может превышать минимального из факторов уверенности участвующих в ней переменных. Наоборот, при использовании связки ИЛИ фактор уверенности посылки всегда оказывается больше максимального из факторов уверенности переменных, участвующих в ней. Это связано с тем, что для истинности посылки со связкой И необходима истинность всех входящих в нее условий, а для связки ИЛИ достаточно истинности одного условия.

При наличии в посылке нечетких переменных может оказаться, что сразу несколько ее значений удовлетворяют условию. В этом случае они объединяются по способу, установленному переменной E.CFCO, т.е. фактор уверенности посылки повышается. Например, если переменная X имеет значения {0 CF 60, 6 CF 40}, то фактор уверенности посылки X<7 будет равен 60 (при E.CFCO= "M").

Значение переменной E.CFVA составляется из двух букв. В качестве каждой из этих букв могут использоваться M, P и A (как и для предыдущих переменных). По умолчанию E.CFVA="PP".

Первая буква в переменной E.CFVA задает способ объединения факторов уверенности посылки и заключения; способ объединения при этом – такой же, как и для соответствующего значения переменной E.CFJO. Пусть, например, имеется правило:

IF X > 0 THEN Y= 0 CF 80

Тогда при стандартном значении переменной E.CFVA и при X=5 CF 70 объединенный фактор уверенности посылки и заключения будет равен 56 (56 = 70 * 80/100). Такой способ объединения факторов уверенности (когда итоговый фактор уверенности посылки оказывается ниже обоих объединяемых факторов) связан с тем, что в данном случае нет полной уверенности ни в том, что посылка правила действительно истинна, ни в том, что при истинности посылки переменная должна получить именно данное значение.

Вторая буква переменной E.CFVA устанавливает способ объединения факторов уверенности значения нечеткой переменной, полученного по нескольким правилам. Если в результате применения правила требуется добавить к набору значений нечеткой переменной такое значение, которое там уже есть, то используется такой же способ объединений, как и для соответствующего значения переменной E.CFCO. Такой способ объединения (когда результирующий фактор уверенности всегда оказывается не меньше максимального из объединяемых факторов) позволяет учесть тот факт, что одно и то же значение временной подтверждается (хотя и с разными степенями уверенности) несколькими правилами.

Если в результате применения правила требуется удалить некоторое значение из набора значений нечеткой переменной, то объединяются факторы уверенности GF1 и 100- CF2; здесь CF1 – фактор уверенности значения переменной до применения данного правила, а CF2 – фактор уверенности значения переменной в заключении правила.

При этом способ объединения устанавливается второй буквой переменной E.CFVA, но для расчета нового фактора уверенности применяется та же формула, что и для соответствующего значения E.CFJO; таким образом, фактор уверенности данного значения переменной снижается.

Пусть, например, имеется правило:

IF V1=1 AND V2=5 THEN G+=2 CF 40; G- =5 CF 20,

т.е. если выполняется посылка, то к нечеткой переменной G следует добавить значение 2 с фактором уверенности 40. Предположим, что в момент обращения к правилу

V1 = 1 CF 30, V2=-5 CF 50, G= {2 CF 20, 5 CF 90}.

Пусть E.CFJO="M", E.CFCO="M", E. CFVA= "MP". Тогда фактор уверенности посылки равен 30 (как минимум из 30 и 50) в соответствии со значением переменной E.CFJO; объединенный фактор уверенности посылки и заключения для G=2 равен 30 (минимум из 30 и 40), а для G=5 – 20 (минимум из 30 и 20) согласно первой букве переменной E.CFVA. Переменная G получит значение 2 с фактором уверенности 44, где 44 = 20 + 30 – 20*30/100; таким образом, ее фактор уверенности увеличится, так как значение, равное 2, подтверждается несколькими (по крайней мере двумя) правилами. Значение 5 для переменной G сохранится, но его фактор уверенности будет равен 72, где 72 = 90 * 80/100, 80 – 100 – 20; таким образом, фактор уверенности для этого значения снижается. В результате применения правила переменная G будет иметь значения {2 CF 44, 5 CF 72}.

1. 3. ЛОГИЧЕСКИЙ ВЫВОД
1.3.1. ОСНОВНЫЕ ВИДЫ ЛОГИЧЕСКОГО ВЫВОДА

В ЭС используется прямой и обратной методы логического вывода знаний.

Прямой вывод. При использовании прямого логического вывода правила просматриваются в прямом направлении (от посылки к заключению). Если посылка правила верна, оно включается, т.е. выполняются действия, указанные в заключении. Если посылка неверна или в ней имеются неизвестные переменные, оно не включается, и происходит обращение к другому правилу. Порядок выбора правил может контролироваться пользователем; по умолчанию правила обычно рассматриваются в порядке их следования. Процесс продолжается, пока не будет определено значение переменной цели или не будет установлено, что определить его невозможно.

Обратный вывод. В системе GURU применяется обратный логический вывод. При использовании обратного логического вывода система для нахождения значения переменной цели просматривает правила в обратном направлении (от заключения правила к его посылке). Система находит все правила, в заключении которых может быть определено значение переменной цели. Одно из этих правил (по умолчанию – первое по порядку, но пользователем может быть установлено и другое) выбирается для рассмотрения. Выбранное правило просматривается системой до посылки, и, если посылка верна, оно включается, в результате чего переменная цели получает значение. Если посылка неверна, это правило не включается, и происходит обращение к следующему правилу, в котором может быть определена цель. Если такого правила нет, то система делает вывод о том, что найти значение переменной цели невозможно, и работа с правилами прекращается.

Если посылка выбранного правила содержит одну или несколько неизвестных переменных, система рассматривает такую переменную, как временная цель, и пытается определить ее значение, аналогично тому, как это делается для основной цели (т.е. отыскиваются правила, в которых временная цель может получить значение, проверяются посылки этих правил и т.д.). Процесс повторяется до тех пор, пока не будет найдено значение основной цели или не будет установлено, что найти его невозможно.

1.3.2. СРЕДСТВА УПРАВЛЕНИЯ ЛОГИЧЕСКИМ ВЫВОДОМ

Система GURU имеет ряд средств, позволяющих управлять логическим выводом, т.е. изменять последовательность рассмотрения правил, составляющих ЭС. Основными средствами управления логическим выводом являются следующие:

- последовательность рассмотрения конкурирующих правил;
- стратегия оценки посылки правила;
- пороговое значение фактора уверенности;
- степень точности при выводе значения неизвестной переменной.

Последовательность рассмотрения конкурирующих правил (т.е. правил, в заключении каждого из которых может быть определена переменная цели) определяется переменной среды E.SORD. Эта переменная может принимать следующие значения:

- F – рассмотрение правил в порядке их расположения в наборе;
- C – по минимуму стоимости;
- P – по приоритету;
- И – по максимальному значению фактора уверенности для текущей цели;
- U – по минимальному количеству неизвестных переменных в посылке правила;
- R – случайный порядок рассмотрения.

По умолчанию E.SORD = "F".

Пусть, например, имеется следующий набор правил;

```
RULE: R1
COST: 45
IF X=1 THEN Y=2 CF 40
```

```
RULE: R2
COST: 40
IF X=2 THEN Y=5 CF 60

RULE: R3
COST: 30
IF (X<>2) AND (X<>1) THEN Y=5 OF 80

RULE: R4
COST: 10
IF Y=2 THEN Z=0 CF 30

RULE: R5
COST: 70
IF Y=5 THEN Z=I CF 90

RULE: R6
COST: 20.
IF Y=0 THEN Z=100 CF 80
```

Переменной цели здесь является Z. Значение переменной X задается в последовательности инициализации. Значения всех управляющих переменных будем считать стандартными.

Пусть задано X=2 с фактором уверенности 100. Проанализируем порядок обращения к правилам при различных значениях E.SORD. Во всех приведенных ниже примерах результат будет один и тот же (Z= 1 CF 54), но порядок рассмотрения правил при этом оказывается различным.

При **E.SORD="F"** сначала рассматривается правило R4 (первое по порядку, в заключении которого может быть определена цель). Неизвестная переменная в его посылке (Y) становится временной целью. Для определения его значения рассматривается правило R1. Оно не включается (посылка ложная). Рассматривается следующее по порядку правило, в котором определяется временная цель Y (R2). Оно включается, и Y получает значение 5 с фактором уверенности 60. Таким образом, посылка правила R4 оказывается ложной. Рассматривается следующее правило с переменной цели в заключении (R5). Оно включается, и Z получает значение 1 CF 64.

При **E.SORD="H"** первым рассматривается правило R6 (как имеющее максимальный фактор уверенности для переменной Z). Для поиска неизвестной переменной Y первым рассматривается правило R3. Оно не включается, и рассматривается правило R2 (как второе по значению фактора уверенности для Y). Для этого правила посылка оказывается истинной, и Y получает значение 5 с фактором уверенности 60. Посылка правила R5 оказывается истинной, и переменная Z получает значение.

При **E.SORD="C"** первым рассматривается правило R4 (так как его стоимость минимальна из стоимостей правил R4, R6 и R6). Для определения переменной Y сначала рассматривается правило R3 (оно не включается), затем – R2 (оно включается; результат – Y = 5 CF 60). Таким образом, посылка правила R4 оказывается ложной, и рассматривается следующее по стоимости правило, содержащее переменную цели (R6). Оно также не включается, и рассматривается правило R5. В этом правиле Z получает значение.

Если в этом наборе правил изменить правило R1 следующим образом:

IF (X=1) AND (L>0) THEN Y=2 CF 40,

а переменную L задавать в каком-либо другом правиле или в последовательности поиска, то при **E.SORD="U" и X=2 CF 100** сначала рассматривается правило R4, затем – R2 (так как в R1 есть одна неизвестная переменная, а в R2 и R3 ее нет). Правило R2 включается, R4 не включается, и рассматривается правило R5, где Z получает значение.

Стратегия оценки посылки правила. *Другим средством управления порядком рассмотрения правил является выбор стратегии оценки их посылок. Эта стратегия устанав-*

ливается переменной среды E.TRYP (для набора правил в целом) или элементом Try (для отдельных из них). Здесь используются три значения:

- S (строгая проверка);
- P (терпеливая);
- E (эталонная).

По умолчанию E.TRYP="S" (как и параметры Try для всех правил).

При E. TRYP="S" система, обнаружив в посылке правила неизвестную переменную, пытается определить ее значение, рассматривая правила, в которых она может быть определена (в порядке, установленном переменной E.SORD). При этом могут рассматриваться заново правила, уже рассмотренные ранее для поиска данной переменной (если это требовалось в других правилах). Система пытается определить значения всех неизвестных переменных, даже если из уже найденных значений можно сделать вывод о том, истинна или ложна посылка. Если хотя бы одну переменную определить не удается, правило не включается.

При E.TRYP="P" система также пытается определить значения всех неизвестных переменных, имеющихся в посылке правила. После выполнения поиска значений для всех неизвестных переменных производится проверка посылки. Если найденных значений достаточно для истинности посылки, то правило включается, даже если некоторые переменные остались неизвестными (это возможно, например, при использовании связки OR).

При E.TRYP="E" система, обнаружив неизвестную переменную, пытается определить ее значение, а по окончании поиска выполняет оценку посылки. Если по имеющимся значениям уже можно сделать вывод об истинности или ложности посылки, то поиск других неизвестных переменных не выполняется.

Пусть, например, имеется правило:

```
IF (X=1) OR (L>0) THEN Y=2
```

Если задано X=1, то при E.TRYP="E" поиск переменной L не выполняется, и правило включается. При E.TRYP="P" поиск значения L выполняется, однако правило включается независимо от его результатов (даже если установить значение L не удается). При E.TRYP="S" поиск значения L выполняется, и если его не удается найти, то правило не включается.

Порог уверенности и точности переменной. Пороговое значение фактора уверенности устанавливается переменной среды E.UNKN. По умолчанию E.UNKN=20. Переменная, у которой максимальный фактор уверенности ниже этого порога, рассматривается как неизвестная; это может привести к изменению порядка рассмотрения правил. Пусть, например, имеется правило:

```
IF X=1 THEN Y=8
```

Пусть задано X=1 CF 15. При стандартном значении порога уверенности это правило не включается, и переменная Y остается неизвестной. Если установить, например, E.UNKN=10, то правило включается.

Порядком рассмотрения правил можно также управлять, изменяя степень точности при выводе значения неизвестной переменной. Для этого используется переменная среды E.RIGR (для всего набора правил) или элемент описания переменной Rigor (для отдельных переменных). Используются три значения:

- M (минимальный поиск);
- A (поиск с использованием всех правил);
- C (тщательный поиск).

По умолчанию используется минимальный поиск.

При E.RIGR="M" поиск значения переменной выполняется до тех пор, пока не будет найдено ее значение, превышающее пороговый фактор уверенности, установленный переменной E.UNKN (или пока не будет установлено, что переменную определить невозможно). После этого другие правила, в заключении которых определяется данная переменная, не рассматриваются.

При E.RIGR="C" после нахождения значения переменной с фактором уверенности, превышающим пороговый, просматриваются последующие правила, в которых эта переменная может получить значение. Однако включаются только те из них, у которых для установления истинности посылок не требуется вывод значений переменных, содержащихся в посылках.

При E.RIGR="A" рассматриваются все правила, в которых может быть определена данная переменная, даже если она уже имеет значение. При этом вывод значений одних и тех же переменных может выполняться многократно.

Рассмотрим, например, порядок вывода для рассмотренного выше набора из шести правил при X=2 и стандартных значениях переменных среды.

Для E.RIGR="M" этот порядок рассмотрен выше (в примере с E. SORD="F").

Для E.RIGR="C" сначала рассматривается правило R4. Переменная Y становится временной целью, и для ее поиска рассматривается правило R1 (не включается), R2 (включается, и Y получает значение 5 с фактором уверенности 60) и R3 (не включается). Так как Y=5, правило R4 не включается, и рассматривается R5. Поиск значения переменной Y повторяется в правилах R1 и R3 (т.е. только в тех, которые не включались); они не включается, и значение переменной Y не изменяется. Правило R5 включается, и переменная Z получает значение (Z=I CF 54). Хотя переменная Z уже известна, рассматривается также правило R6, однако вывод значения переменной Y не повторяется, и R6 не включается.

Для E.RIGR="A" вывод выполняется аналогично рассмотренному для E.RIGR="C", но после обращения к правилу R6 система снова пытается вывести значение переменной Y, обращаясь для этого к правилам R1 и R3; однако они не срабатывают, и значения Y и Z не изменяются.

1.4. ПОДСИСТЕМА ОБЪЯСНЕНИЯ

Запрос во время консультации. Во время консультации может создаться впечатление, что действия, выполняемые машиной логических выводов, не имеют отношения к проблеме. Это возможно потому, что пользователь не знает, как происходит внутренний процесс аргументации. Если пользователь действительно не понимает, почему от него требуют той или иной информации, он может отреагировать, используя (CTRL-I). В этом случае он увидит на экране дисплея текущее обрабатываемое правило. После нажатия ENTER это объяснение исчезает, и он может ввести ответ.

Запрос после консультации. После консультации с набором правил пользователь может попросить систему объяснить, какие правила и переменные использовались. Для этого применяются две команды:

- HOW – выдает переменные, которые использовались;
- WHY – объясняет правила, которые использовались.

Перечень информации, которая дает использование этих команд следующий:

- HOW – выдает значение переменной цели, правило или правила, с помощью которых была определена цель.
- HOW "имя переменной" – выдает значение или значения с указанной переменной.
- HOW "число" – выдает значение или значения переменной с порядковым номером, заданными этой переменной в наборе правил.
- WHY – воспроизводит на экране дисплея объяснение (REASON) и переменные, которые требовались для правила, выполняющегося последним. Переменные отображаются с порядковый номером, который можно потом использовать в команде HOW (см. выше).
- WHY "имя правила" – воспроизводит на экране дисплея объяснение (REASON) и переменные, необходимые для данного правила.
- WHY "число" – воспроизводит на экране дисплея объяснение и переменные, необходимые для REASON правила с указанным порядковым номером в наборе правил.

Для того, чтобы объяснить процесс аргументации, необходимо использовать HOW и WHY совместно.

1.5. КОМАНДНЫЙ РЕЖИМ ВЗАИМОДЕЙСТВИЯ С GURU

Для работы пользователя в режиме командной строки необходимо сначала в главном меню выбрать элемент ***Change Environment***, а затем в субменю – элемент ***Command Mode***.

Основные команды, используемые в командной строке, следующие:

BUILD – создать набор правил из командной строки.
COMPILE создать на основе текста набора правил ЭС, которую можно выполнить.
CONSULT – запуск ЭС для консультации.
RUN – запуск внешней программы (*.EXE и *.COM), работающие под управлением MSDOS. Система "GURU" возобновляет обработку после того, как внешняя программа выполнена.
DIR – просмотр существующих файлов в директории устройства.
LET – присвоение значения переменной.
INPUT – ввод данных в переменную.
OUTPUT – вывод сообщения на терминал.
PERFORM – запуск программы "GURU".
RETURN – завершение выполнения процедуры и обеспечение возврата в вызывающую программу или в командный режим.
WAIT – останов выполнения программы до тех пор, пока пользователь не нажмет какую-либо клавишу
WHILE < условия > DO < утверждения > ENDWHILE создание цикла, состоящего из других команд "GURU". Цикл обрабатывается до тех пор, пока не будут удовлетворены заданные условия.
TEST – проверка значения специфицированного выражения для определенного варианта.

```
TEST < выражение >
CASE < выражение – 1: > < утверждение – 1 >
CASE < выражение – 2: > < утверждение – 2 >
:
OTHERWISE: < утверждение >
ENDTEST.
```

IF-THEH-ELSE – реализация одного набора команд, если удовлетворяются указанные условия или другого набора команд в противном случае.
CONTINUE – останов обработки части DO команды WHILE-DO и повторная оценка условия WHILE.
BREAK – останов обработки DO-WHILE или обеспечивает выход из TEST.
CLEAR – очистка экрана.
BYE – выход из командного режима.
HOW переменная – вывод информации о том, как система определила значение переменной.
WHY правило – вывод информации об обработке правила.
HELP – вызов подсказки.
SHOW ENVIRONMENT – вывод значений всех переменных среды.

Команды текстового редактора

TEXT <имя файла> – запуск текстового редактора "GURU";
READ <имя файла> – завершение обработки текущего текста и начало обработки другого текстового файла без остановки и повторного вызова текстового редактора;
TOP – передвижение курсора в верхнюю часть текущего текста;
BOTTOM – передвижение курсора в нижнюю часть текущего текста;
GOTO <позиция> – передвижение курсора в определенную строку текста;
SEARCH – поиск определенной группы символов в тексте;
INSERT <имя файла> – вставка содержимого файла в текущий текст;
PRINT – печать текста;
QUIT – выход из текстового редактора без сохранения изменений;
STOP – выход из текстового редактора с сохранением;
<CTRL-!> – помощь.

Перечень и формат команд для обеспечения командного режима взаимодействия пользователя с оболочкой приведен в приложении 5.

1.6. ДИАЛОГОВЫЙ РЕЖИМ ВЗАИМОДЕЙСТВИЯ С GURU

1.6.1. ОБЩИЕ ПОЛОЖЕНИЯ

Диалоговое взаимодействие является основным режимом взаимодействия разработчика ЭС с оболочкой GURU. Общая схема взаимодействия представлена на рис.1.2. Методика взаимодействия пользователя с оболочкой включает последовательность следующих этапов:

1. Запуск оболочки;
2. Определение исходного файла для организации взаимодействия;
3. Выбор режима работы:
   - *Создание текста программы ЭС* (новой или редактирование существующей).
   - *Режим использования программы ЭС* (решения задачи).
4. Завершение взаимодействия с оболочкой.

Запуск оболочки. Для запуска экспертной оболочки GURU необходимо выполнить следующие действия:

- Копировать в свой сетевой каталог из системного каталога папку с именем ***GURU***.
- Запустить на выполнение модуль ***GURU.EXE***.

Определение исходного файла для организации взаимодействия. В рамках данного действия определяется либо продолжение существующего процесса взаимодействия, либо определяется новое взаимодействие. После запуска ЭО на экране появится сообщение:

RESUME PREVIOUS SESSION:

оно требует от пользователя ответа – продолжить предыдущее взаимодействие с оболочкой (ответ ***Y***) или начать новое (ответ ***N***).

В случае первого или нового взаимодействия с оболочкой выводит следующее сообщение:

NEW SESSION NAME:

В качестве ответа на это сообщение пользователь должен задать ***имя файла***, в котором будет храниться создаваемый в рамках оболочки исходный текст программы ЭС.

Выбор режима работы пользователя. После выбора режима взаимодействия необходимо определить режим работы. Определено два основных режима работы:

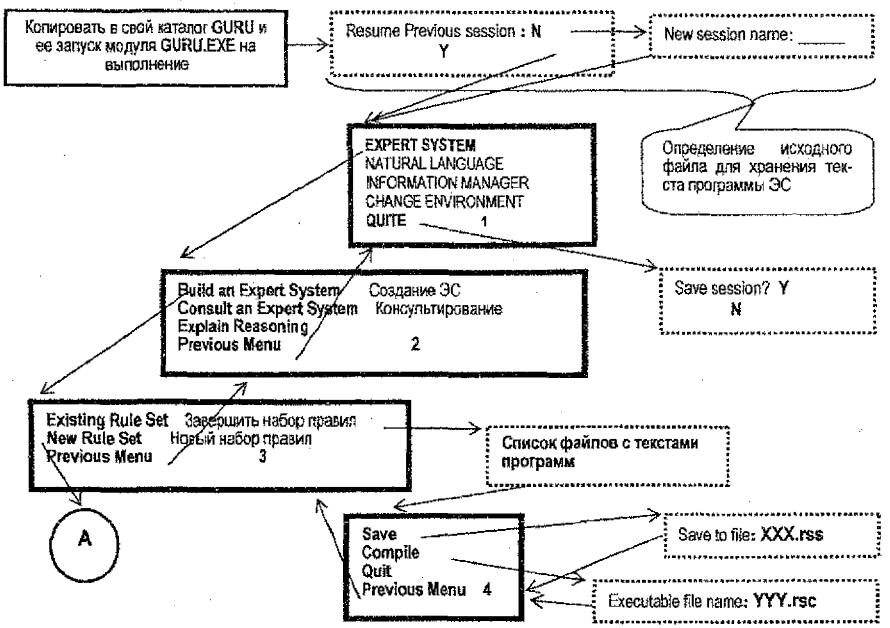
- ***Создание программы ЭС***, т.е. подготовка исходного текста программы ЭС – выбор ***Build an Expert System.*** Процесс создания разделов текста программы ЭС представлен на рис.1.3.
- ***Консультирование*** или решение задачи ЭС – выбор ***Consult an Expert System.*** Для режима консультирования необходимо выбрать файл с исходным текстом готовой программы ЭС, откомпилировать его и запустить на выполнение. Процесс консультирования заключается во вводе необходимых исходных данных пользователем для ЭС и выдача решения программой ЭС.

Завершение взаимодействия с оболочкой. Для завершения работы ЭС необходимо сохранить созданные файлы ЭС и с помощью команды ***QUIT*** завершить взаимодействие с оболочкой.

Схема диалогового взаимодействия с оболочкой. Обобщенная схема диалогового взаимодействия пользователя с оболочкой представлена на рис.1.2.

В главном меню оболочки (см. рис.1.2, ***блок 1***), выводимом на экран сразу же после ее запуска, для работы с ЭС предназначен его первый пункт ***Expert System***. После выбора этого пункта оболочка предоставляет следующие возможности (***блок 2***):

- ***создание текста программы*** ЭС – пункт ***Build Expert System***;
- ***консультирование*** с существующей ЭС – пункт ***Consult Expert System***;
- ***объяснение процесса работы*** ЭС по окончании консультации – пункт ***Explain Reasoning***;
- возврат в предыдущее меню – пункт ***Previous Menu*** или нажатие клавиши ***Esc***.



Примечание: Для ссылок на пункты меню оболочки, которые приведены на этом рисунке, используется нумерация блоков в правом нижнем углу (блок 1, блок 2, и т.д.)

Рисунок 1.2 – Обобщенный сценарий взаимодействия пользователя с оболочкой

Для построения или изменения текста программы ЭС необходимо выбрать элемент меню ***Build Expert System*** *или ввести в командной строке команду* ***BUILD***. *После этого* оболочка GURU предоставляет пользователю следующие возможности (***блок 3***):

- ***создание нового текста программы*** – пункт ***New Rule Set;***
- ***использование существующего текста программы*** – пункт ***Existing Rule Set;***
- возврат в предыдущее меню – пункт ***Previous Menu*** или нажатие клавиши ***Esc***.

В первом случае система запрашивает имя нового файла (под этим именем файл будет сохранен), а во втором – выводит на экран список имен уже имеющихся файлов для выбора одного из них. Затем на экран выводится меню, в рамках которого формируются разделы программы ЭС (см. рис.1.3).

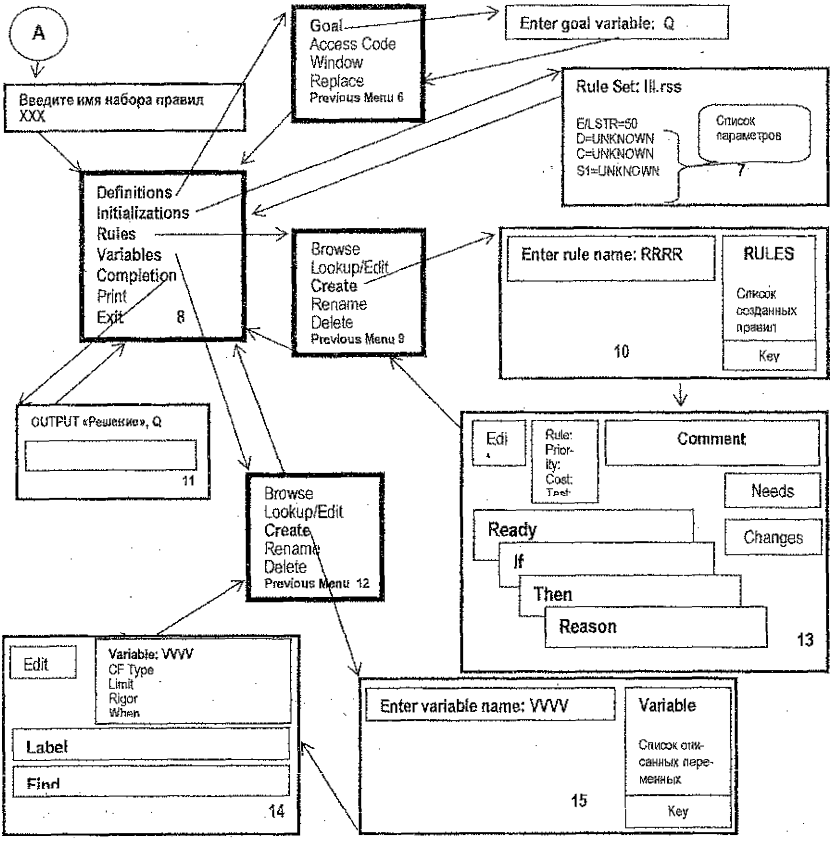


Рисунок 1.3 – Структурная схема сценария построения разделов программы ЭС

По завершению действий, связанных с формированием текста программы ЭС, оболочка предоставляет следующие возможности (***блок 4***):

- ***сохранить текст*** программы ЭС в файле (***Save***);
- ***компилировать текст*** программы ЭС (***Compile***);
- завершить работу и вернуться в меню более высокого уровня (***Quit***);
- возврат в предыдущее меню – пункт ***Previous Menu*** или нажатие клавиши ***Esc***.

Исходный текст программы ЭС сохраняется в файле с указанным ранее именем и с расширением ***RSS***, а откомпилированная программа – в файле с тем же именем и расширением ***RSC***. Откомпилированный файл может затем выполняться или применяться для консультаций с ЭС. При попытке закончить работу без сохранения текста программы ЭС оболочка выдает соответствующее предупреждение.

Как было определено в п.1.2.8, в состав текста программы ЭС входит пять разделов (цели, инициализации входных переменных, завершающих действий, правил вывода, входных параметров). Для организации подготовки текста программы в рамках оболочки используется система меню. Каждый из разделов имеет свой сценарий построения, основные возможности которых представлены на рис.1.3. Каждый из разделов программы можно создавать и редактировать независимо.

Для выбора возможности по формированию отдельных разделов программы ЭС используется меню (рис. 1.3., ***блок 8***), в котором пользователю предоставляются следующие возможности:

- ***Создание целевой переменной*** – пункт ***Defenitions***;
- ***Инициализация переменных*** – пункт ***Initializations***;
- ***Формирование правил вывода*** – пункт ***Rules***;
- ***Определение переменных*** – пункт ***Variables***;
- ***Задание завершающих действий*** – пункт ***Completion***;
- Печать – пункт ***Print***;
- Выход – пункт ***Exit***.

Сценарии диалогов для создания отдельных разделов программы на входном языке оболочки рассмотрены в п.1.6.2 – п.1.6.6.

1.6.2. РАЗДЕЛ ЦЕЛИ

В этом разделе определяется целевая переменная ЭС. В качестве такой переменной используется целевая переменная, которая определена на этапе структурирования знаний, а затем уточнена в процессе формализации [3].

Цель – это переменная, значение которой оболочка GURU стремится определить во время консультации с ЭС. Определение значения переменной цели означает завершение консультации. Указание цели и ее использование в тексте программы ЭС обязательно. ***Раздел цели ЭС*** имеет следующую структуру:

GOAL: <***переменная цели***>— оператор определения целевой переменной;

Пример раздела цели в тексте программы:

GOAL: POSITION

В приведенном примере указано, что целевой переменной для ЭС является переменная ***POSITION***. Кроме этого, целевая переменная должна быть включена в состав переменных, которые определяются в разделе инициализации.

Формирование раздела цели под управлением оболочки реализуется в рамках сценария, который приведен на рис.1.4. Переходы между пунктами отдельного меню выполняются с помощью клавиш ↑ и ↓. Для создания раздела цели необходимо последовательно выбрать ***пункт Definition (блок 8) – пункт Goal (блок 6).*** Далее оболочка выдаст запрос <***Enter goal variable***>, на который пользователь должен ввести имя целевой переменной. После ввода переменной и нажатия клавиши ввода (***ENTER***) оболочка формирует текстовую конструкцию, пример которой аналогичен приведенному выше примеру. Текст сформированного раздела цели автоматически добавляется в файл, в котором хранится текст программы ЭС. На рис.1.4 в качестве целевой переменной приведена переменная с именем ***POSITION***.

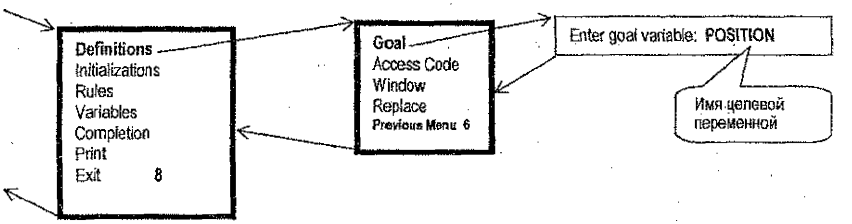


Рисунок 1.4 – Процесс формирования раздела цели

1.6.3. РАЗДЕЛ ИНИЦИАЛИЗАЦИИ

Раздел инициализации переменных ЭС – это последовательность команд, выполняемых оболочкой при обращении к набору правил до рассмотрения каких-либо правил из программы. В этой последовательности может выполняться, например, ввод или задание начальных значений рабочим переменным, установка значений переменных среды. Этот раздел начинается ключевым словом ***INITIAL*** и может включать последовательность команд. Данный раздел имеет следующую структуру:

INITIAL: ***<последовательность операторов>*** — включает последовательность операторов по разным видам обработки переменных (ввод, инициализация и пр.), которые необходимо выполнить до начала консультации.

Пример раздела инициализации в тексте программы ЭС:

```
INITIAL:
        E.LSTR=30
        POSITION=UNKNOWN
        DEGREE=UNKNOWN
        QUALIFY=UNKNOWN
        DISCOVERY=UNKNOWN
        EXPERIENCE=UNKNOWN
```

В данном примере приведено изменение значения переменной среды (установка максимальной длины строки при выводе на экран равной 30 – ***E.LSTR = 30***) и перечислено пять переменных (***POSITION, DEGREE, QUALITY, DISCOVERY, EXPERIENCE***), которым присвоены значения ***UNKNOWN*** (что означает неизвестное значение).

Создание операторов раздела инициализации переменных под управлением оболочки реализуется в рамках сценария, который приведен на рис.1.5. Для создания этого раздела необходимо последовательно выбрать ***пункт Initializations (блок 8)*** , затем в окне ***(блок 7)*** формируется текст операторов, которые должны войти в состав данного раздела. На рис.1.5 в ***блоке 7*** приведено описание операторов (команд) для раздела инициализации, аналогичное рассмотренному выше примеру.

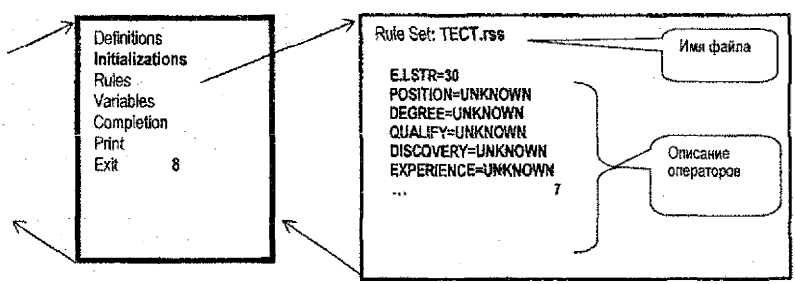


Рисунок 1.5 – Сценарий создания раздела инициализации переменных

После формирования необходимых команд и нажатия клавиши ***Esc*** оболочка (на основе информации из ***блока 7***) формирует текст раздела инициализации переменных и автоматически добавляет его в текст программы ЭС, которая хранится в текстовом файле. Имя файла (см. рис.1.5 – ***TEST.RSS***) с текстом программы отображается оболочкой в верхней части ***блока 7***.

1.6.4. РАЗДЕЛ ЗАВЕРШАЮЩИХ ДЕЙСТВИЙ

Раздел завершающих действий ЭС предназначен для определения последовательности команд, выполняемых по завершении (консультаций с ЭС) процесса вывода решения из набора правил. Эта последовательность может использоваться, например, для вывода значения переменной цели или других переменных. Данный раздел начинается ключевым словом ***DO*** и имеет следующую структуру:

DO: ***<последовательность операторов>*** – включает последовательность команд для разных видов обработки, которые необходимо выполнить после консультации (нахождения значения целевой переменной), т.е. после завершения выполнения программы ЭС.

Например,

DO: OUTPUT "РЕШЕНИЕ ", POSITION

В этом примере представлен вывод значения целевой переменной ***POSITION***, после вывода на экран текста «**РЕШЕНИЕ**».

Формирование раздела завершающих действий под управлением оболочки реализуется в рамках сценария, который приведен на рис.1.6.

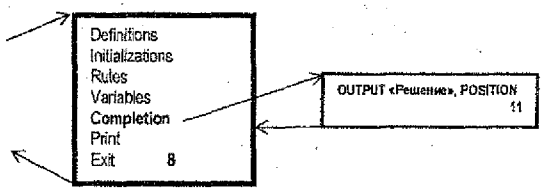


Рисунок 1.6 – Сценарий определения раздела заключительных действий

Для создания данного раздела необходимо последовательно выбрать ***пункт Completion (блок 8)***, а затем в окне ***(блок 11)*** набрать текст операторов. На рис.1.6 в качестве примера приведен перечень операторов для раздела завершающих действий, аналогичный рассмотренному выше примеру. Текст раздела завершающих действий формируется на основе ***информации из блока 11,*** который оболочка автоматически добавляет в программу ЭС, хранящуюся в текстовом файле.

1.6.5. РАЗДЕЛ ПРАВИЛ

Раздел правил является основным разделом программы ЭС и состоит из совокупности правил вывода. Структура раздела правил имеет следующий вид:

< правило вывода 1 > – определяет начало раздела правил;

...

<правило вывода N> – определяет N-е правило вывода;

Правила вывода представляют собой операторы вида ***IF ... THEN ...*** Основными элементами любого правила являются посылка (***IF***) и заключение (***THEN***).

Посылка представляет собой логическое выражение. Это может быть единственное выражение или группа выражений, соединенных логическими операторами ***OR*** (Или) и ***AND*** (И). В ней могут использоваться константы, переменные различных классов, операторы отношения (<>,<-,>-), арифметические операторы, числовые и строковые функции.

Заключение представляет собой последовательность одной или нескольких команд GURU, например, команд присваивания значений переменным, ввода-вывода переменных и т.д.

Отдельное правило вывода ЭС структурно может включать пять компонентов, каждый из которых начинается с новой строки:

RULE: [Имя или номер правила вывода]
[READY: <Последовательность команд>]
IF: <Условие>
THEN: <Действие>
[REASON: Комментарий для вывода подсистемой объяснения]

При описании структуры правила использованы следующие обозначения:

[READY: <Последовательность команд>] – Последовательность готовности **(Ready)** – одна или несколько команд, выполняемых при обращении к правилу до проверки его посылки на истинность. Этот элемент обычно используется для присваивания или ввода значений переменных, используемых в правилах. Кроме того, его можно использовать для отслеживания порядка обращения к правилам, указав здесь команду вывода, например, следующую: **OUTPUT "Обращение к правилу 1"**

<Условие> – это *логическое выражение над переменными ЭС, при истинности которого выполняется* указанное в правиле действие;

<Действие> – это обычно либо вычисление значения выражения и присвоение его конкретной переменной, либо присвоение определенных значений конкретным переменным;

[REASON: Комментарий для вывода подсистемой объяснения] – это строка комментария, который выводится на экран во время работы с подсистемой объяснений после завершения решения задачи.

Рассмотрим структуру отдельного правила из примера, приведенного на рис.1.9:

RULE: R1
READY: OUTPUT "RR1"
IF: (DEGREE="ДА") AND (GRADE>3.5)
THEN: QUALIFY= "ДА"

В первой строке определено имя правила – ***R1***, во второй строке – определен вывод символьной константы «***правило R1***», при отладке ЭС. Третья и четвертая строки – это само правило, которые можно сформулировать следующим образом:

*Если значение переменной **DEGREE** равно «Да» и значение переменной **GRADE** больше **3.5**, то переменной **QUALIFY** присвоить значение «Да».*

Формирование раздела правил под управлением оболочки реализуется в рамках сценария, который приведен на рис.1.3. Для работы непосредственно с набором правил вывода следует вызвать из меню элемент ***Rules***, который выводит меню (***блок 9***), обеспечивающее работу с правилами. После этого можно просматривать и редактировать правила последовательно (***Browse***) или обратиться к конкретному правилу по имени (***Lookup/Edit***), создавать новые правила (***Create***), переименовывать (***Rename***) и удалять (***Delete***) уже существующие.

При обращении к элементу меню ***Browse*** можно работать последовательно со всеми правилами (***All***) или обращаться к правилам, содержащим конкретную строку (***String***). При работе с правилом имеется возможность перейти к следующему (***Next***) или к предыдущему (***Prior***), приступить к редактированию текущего правила (***Edit***) или завершить работу с набором правил (***Return***).

Процесс создания правила осуществляется в рамках блоков 10 и 13 (см. рис.1.7). Описание различных элементов правил выполняется с помощью выводимой на экран системы окон. Переходы между окнами выполняются с помощью клавиш ***PgUp*** и ***PgDn***, а завершение формирования текущего правила – нажатием клавиши ***Esc***.

В ***блоках 10*** и ***13*** формируются необходимые составные компоненты правила (***Ready, If, Then, Reason***), а также определяются другие необязательные элементы (***приоритет, стоимость, проверка, комментарий, список потребностей, список изменений***). Отдельные компоненты правила формируются в отдельных окнах, изображенных на рис.1.7.

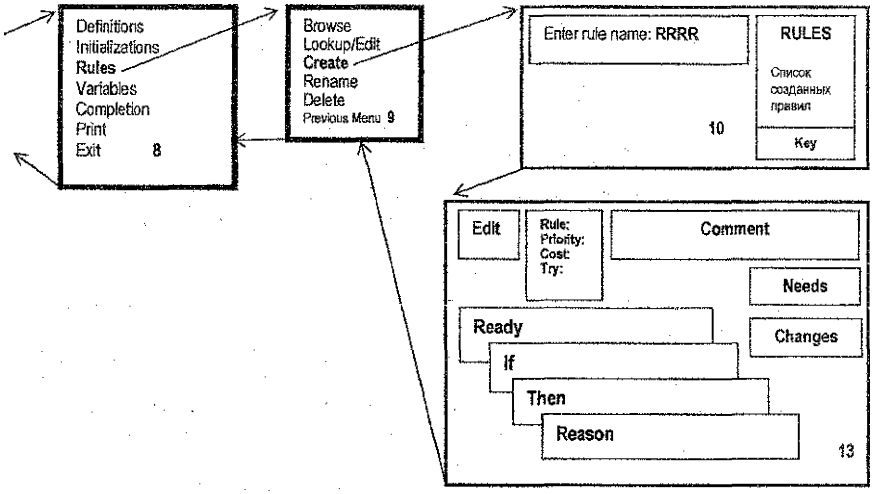


Рисунок 1.7 – Сценарий определения раздела правил

Описание и интерпретация необязательных элементов правила, представленных в блок 13 на рис.1.7., следующие.

- ***Приоритет (Priority)*** – целое число от 1 (низший приоритет) до 100 (высший), указывающее на приоритет правила.
- ***Стоимость (Cost)*** – целое число от 1 (низшая) до 100 (высшая), указывающее на стоимость действия правила.
- ***Проверка (Try)*** – код, устанавливающий стратегию оценки посылки правила. Эта стратегия устанавливается переменной среды ***E.TRYP*** для всех правил, но для конкретного правила может быть переопределена с помощью данного кода.
- ***Комментарий (Comment)*** – произвольный текст, используемый для пояснения смысла или назначения правила; его указание необязательно.
- ***Список потребностей (Needs)*** – список переменных, используемых для оценки посылки. Вводить этот список необязательно. Кроме того, в нем перечисляются переменные, используемые в заключении правила и определяемые вне набора правил (например, в последовательности инициализации или поиска значения переменной); указание таких переменных в этом списке обязательно.
- ***Список изменений (Changes)*** – список переменных, изменяемых в данном правиле: указание этого списка в большинстве случаев необязательно.

Элементы «приоритет» и «стоимость» могут использоваться для управления процессом вывода решения, осуществляемым на основе набора правил. С помощью переменной среды ***E.SORD*** может быть задана последовательность обработки правил в порядке убывания приоритетов или возрастания стоимости.

1.6.6. РАЗДЕЛ ОПИСАНИЯ ПЕРЕМЕННЫХ

Этот раздел предназначен для описания рабочих (входных и промежуточных) переменных, используемых в правилах. В нем должны быть указаны все переменные задачи (в том числе и переменная цели). Формат описания отдельной переменной следующий:

VAR: **<Имя входной переменной>**
FIND: **<Оператор ввода Имя входной переменной>**
LABEL: **[Комментарий для входной переменной]**

Раздел описания переменных включает перечень определений по одному на каждую входную переменную. При описании входной переменной использованы следующие обозначения:

<Имя входной переменной> – имя, под которым входная переменная используется в ЭС;

FIND: **<Оператор ввода Имя входной переменной>** – конструкция для организации ввода значения входной переменной для решения задачи, это команды (одна или несколько), выполняемые для нахождения значения данной переменной, если оно неизвестно. В простейшем случае это одна команда ввода.

LABEL: **[Комментарий для вывода подсистемой объяснения]** – это комментарий в виде произвольного текста (до 48 символов), выводимый на экран во время работы с подсистемой объяснений.

Например, пусть необходимо определить входную переменную ***GRADE***, означающую средний балл за время учебы в вузе, и ввести ее значение путем вывода на экран сообщения «***Какой средний балл (0-5)***=», и последующего ввода значения с клавиатуры в переменную ***GRADE***. Это описание будет иметь следующий вид:

VAR: **GRADE**
FIND: **INPUT GRADE NUM WITH "Какой средний балл (0-5) ="**
LABEL:**СРЕДНИЙ БАЛЛ ЗА ВРЕМЯ ОБУЧЕНИЯ**

Формирование раздела описания переменных под управлением оболочки реализуется в рамках сценария, который приведен на рис.1.8.

Для создания данного раздела необходимо последовательно выбрать ***пункт Variables (блок 8) – пункт Create (блок 12),*** а затем в ***блок 15 и блок 14*** определяется отдельная переменная для ЭС. В ***блоке 15*** определяется имя переменной (на рис.1.8 это имя ***VVVV***), а в ***блоке 14*** формируются составные компоненты правила (***Label, Find***), а также определяются другие необязательные элементы (***тип фактора уверенности, предел, точность, время поиска***). Отдельные компоненты правила формируются в отдельных окнах, изображенных на рис.1.8.

Текст раздела определения переменных оболочка формирует на основе ***информации из блоков 14 и 15*** и автоматически его добавляет в текст программы ЭС, которая хранится в заданном файле.

Описание и интерпретация необязательных элементов правила, представленных в блок 13 на рис.1.8, следующая.

- ***Тип фактора уверенности (CF Type)*** – двухсимвольный код, устанавливающий способ вычисления фактора уверенности для данной переменной.
- ***Предел (Limit)*** – целое число, указывающее максимальное количество значений, которое может иметь переменная, если она используется как нечеткая переменная; это число не может превышать 255,а также величины, установленной переменной среды ***E.NUMV***.
- ***Точность (Rigor)*** – код, устанавливающий способ вывода значения данной переменной, если оно неизвестно.
- ***Время поиска (When)*** – код, устанавливающий время выполнения последовательности Find.

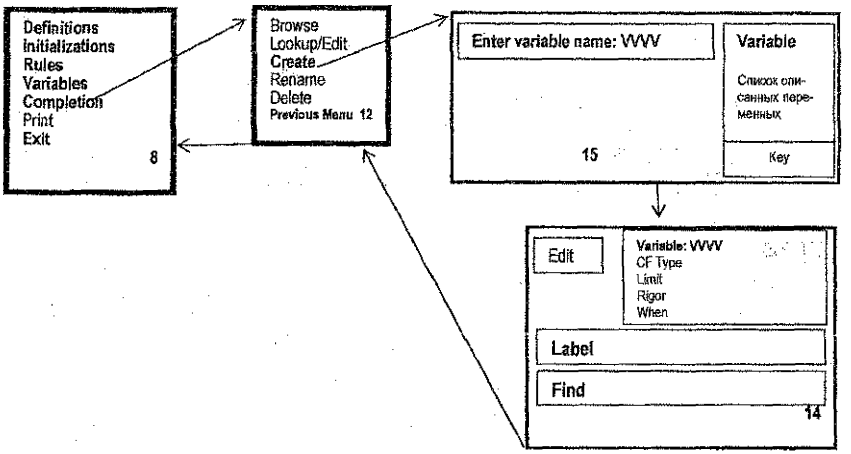


Рисунок 1.8 – Сценарий формирования раздела определения переменных

Характеристики ***When, CF Type, Rigor и Limit*** задаются переменными среды ***E.WHN, E.CFVA, E.RIGR*** и ***E.NUMV*** (соответственно) для всех переменных, имеющихся в наборе правил. Указание этих характеристик для отдельных переменных позволяет переопределить их для данных переменных.

1.7. ПРИМЕР ПРОГРАММЫ ЭКСПЕРТНОЙ СИСТЕМЫ

В качестве примера рассмотрим построение программы ЭС (см. рис.1.9) для задачи принятия решения о приеме на работу сотрудников и в выборе им должности [10].

Предположим, что решение принимается на основе следующих правил:

- если кандидат не имеет высшего образования, то он не может быть принят на работу;
- если кандидат имеет высшее образование, а также имеет разработки, связанные со сферой деятельности данной организации, то его следует принять на должность научного сотрудника;
- если кандидат имеет высшее образование и разработки у него отсутствуют, то следует выяснить у кандидата его средний балл за время учебы, и если он превышает 3,5 балла, то принять на должность инженера-конструктора;
- если средний балл оказывается ниже 3.5, то выяснить у кандидата, каков стаж его работы по специальности, и если он превышает 2 года, то принять на должность инженера по эксплуатации, а если менее 2 лет – отказать в приеме.

Процесс решения задачи целесообразно представить в виде дерева решений. Вершины этого дерева могут соответствовать вопросам (их называют вершинами решений) или логическим выводам. Вершины, содержащие вопросы, всегда связаны выходящими ребрами с несколькими другими вершинами (в зависимости от ответа на вопрос); это могут быть как вершины логических выводов, так и другие вопросы. Логические выводы могут быть окончательными (в данном примере это решения о предлагаемой должности или об отказе в приеме) или промежуточными (например, решение о возможности приема на работу, если кандидат имеет высшее образование). Вершины, соответствующие окончательным выводам, представляют собой концевые вершины дерева.

Дерево решений преобразуется в набор правил. Посылки правил (*IF*) соответствуют вопросам, заключения (части *THEN*) – выводам (в качестве вывода здесь может использоваться не только окончательный, но и некоторый промежуточный вывод или вопрос). Посылка правила содержит все вершины вопросов, находящиеся на пути от начальной вершины (вопроса, задаваемого первым) к выводу, содержащемуся в заключении правила.

Текст программы-прототипа ЭС, которая реализует рассмотренную выше задачу, приведен на рис.1.9. Использованные переменные описаны в табл.1.5.

Таблица 1.5 – Описание переменных ЭС

№ п/п	Обозначение переменной	Назначение переменной	Тип переменной, диапазон значений	Примечание
1	*POSITION*	Решение о предлагаемой должности или об отказе в приеме	Символьная (Отказать, Научный работник, Инженер-конструктор, Инженер по эксплуатации)	Целевая переменная
2	*DEGREE*	Наличие или отсутствие высшего образования	Символьная (ДА или НЕТ)	Входная переменная
3	*DISCOVERY*	Наличие или отсутствие у кандидата разработок по специальности	Символьная (ДА или НЕТ)	Входная переменная
4	*GRADE*	Средний балл за время учебы	Числовая	Входная переменная
5	*EXPERIENCE*	Стаж работы по специальности	Числовая	Входная переменная
6	*QUALIFY*	Промежуточное решение о возможности приема на работу, если кандидат имеет высшее образование		Промежуточная переменная

Алгоритм функционирования программы ЭС следующий. Переменная ***POSITION*** является целевой. Первый вопрос (о высшем образовании) задается в последовательности инициализации. Окончательный вывод (т.е. значение переменной ***POSITION*** по окончании консультации) выдается на экран в последовательности завершения.

Рассмотрим последовательность обработки данного набора правил для случая, когда все переменные среды имеют стандартные значения, а на первый вопрос (о высшем образовании) введен ответ "***ДА***". Сначала рассматривается правило R1, так как оно является первым по порядку, в котором определяется переменная цели. Посылка R1 ложна (***DEGREE="ДА"***), и рассматривается следующее правило, в котором может быть определена цель. Это правило – R4; в его посылке есть неизвестная переменная ***DISCOVERY***. Она становится временной целью, и рассматривается первое правило, в котором она может быть определена (R3).

Предположим, что введен ответ "***НЕТ***". Поиск временной цели ***DISCOVERY*** заканчивается, посылка R4 оказывается ложной, и рассматривается следующее правило, в котором определяется основная цель (R6). В его посылке есть одна неизвестная переменная (***GRADE***), которая становится временной целью, и для ее определения происходит обращение к правилу R5. Его посылка истинна, и оно включается; запрашивается средний балл (значение переменной ***GRADE***). Пусть введено число 4.5, тогда посылка правила R6 оказывается истинной, правило включается, и переменная цели получает значение "***ИНЖЕНЕР-КОНСТРУКТОР***". Обработка правил на этом заканчивается; выполняется последовательность завершения, и на экран выводится ответ: ***ИНЖЕНЕР-КОНСТРУКТОР.***

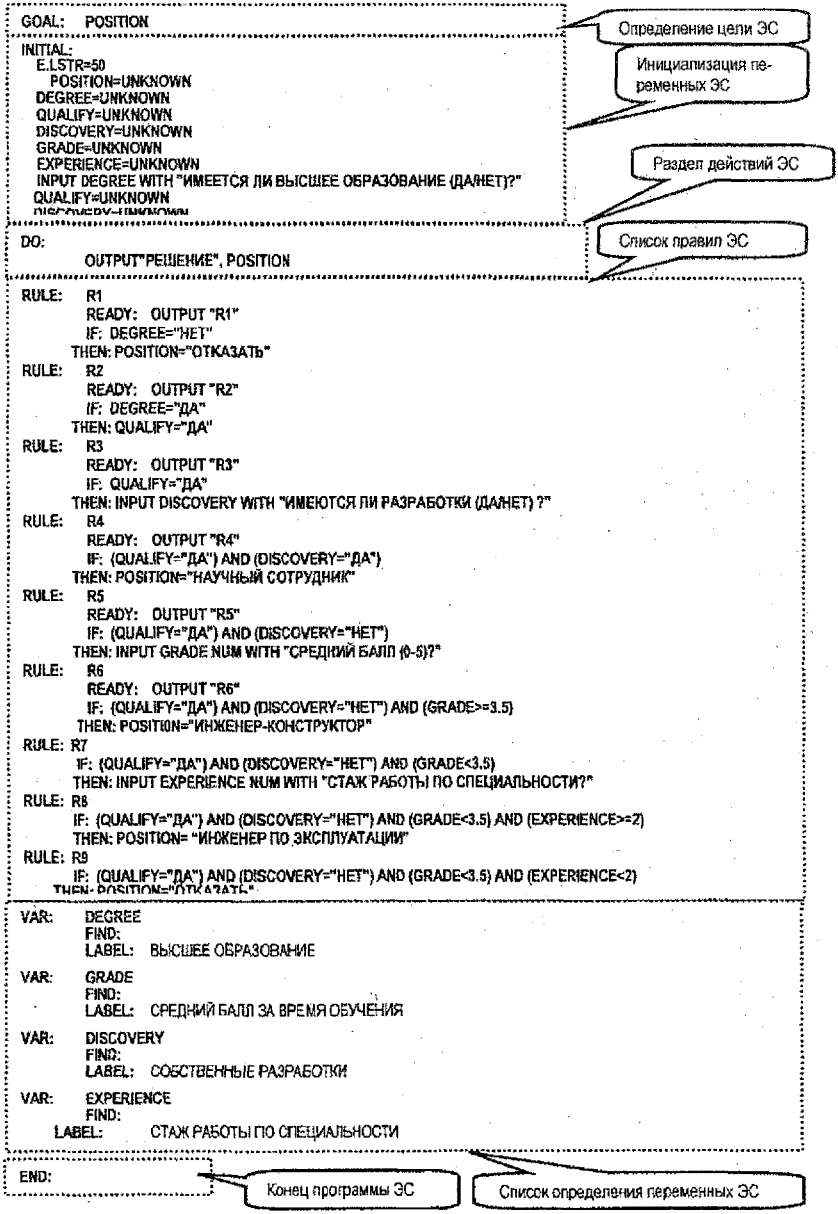


Рисунок 1.9 – Пример исходного текста программы ЭС на входном языке системы GURU

2. РЕАЛИЗАЦИЯ ЭС

Программу-прототип ЭС как программное изделие в рамках оболочки GURU будем рассматривать как отдельный процедурный программный модуль, реализация которого выполняется в рамках ГОСТ единой программной документации (ЕСПД).

Сложность программы-прототипа ЭС зависит от количества и связности правил, составляющих основу ЭС.

Процесс реализации программы ЭС предполагает выполнение следующих видов работ:

- ***Подготовка, проверка и компилирование исходного текста программы*** ЭС на входном языке GURU;
- ***Отладка и тестирование программы*** ЭС;
- ***Разработка документации*** на ЭС;
- ***Испытание и внедрение*** ЭС.

В общем случае разработка и реализация прототипов предполагает итеративный процесс создания ЭС. Это вызвано тем, что модель решения задачи в процессе ее реализации и проверки может быть частично или полностью изменена.

В рамках лабораторного практикума по дисциплине «Технология и инструментальные средства проектирования интеллектуальных систем» будем рассматривать одну итерацию этого процесса.

Проблемы проверки правильности функционирования ЭС. ЭС как объект проверки характеризуется двумя важными аспектами:

1. ЭС является программой, функционирование которой определяется и управляется МЛВ оболочки GURU. Процесс выполнения программы ЭС отличается от выполнения обычных модульных программ (см. п.1.2.11 и п.1.3). Кроме этого, количество и связность правил ЭС увеличивает сложность программы, как изделия, так и время ее отладки и проверки. Для проверки первого аспекта приведена схема на рис.2.1. Основные причины неудач при создании и проверке программы ЭС следующие:
   - Определение переменных ЭС.
   - **Ввод-вывод** можно характеризовать данными, приобретаемыми в ходе диалога с экспертом, и заключениями, предъявляемыми ЭС в ходе объяснений. Методы извлечения знаний могут не давать нужных результатов, так как задавались неправильные вопросы или собрана не вся необходимая информация. Кроме того, вопросы системы могут быть трудными для понимания. Ошибки при вводе могут возникать из-за неудобного для пользователя входного языка. В ряде случаев для пользователя удобен ввод не только символьной информации, но и графической. Выходные сообщения системы могут оказаться не понятными пользователю или эксперту либо потому, что их слишком много, либо потому, что их слишком мало. Они могут быть организованы, упорядочены способом, не понятным или не удобным пользователю или выражены на не подходящем для пользователя уровне абстракции с использованием непонятной лексики.
   - **Тестовые примеры** часто оказываются вне ПрО, для которой разработана ЭС. Иногда тестовые примеры находятся в ПрО, но не охватывают всю ПрО. При подготовке тестовых примеров целесообразно классифицировать их по подпроблемам ПрО, выделяя стандартные случаи, определяя границы трудных ситуаций и т.д.

2. **Правила вывода и управляющие стратегии**. Основные ошибки в правилах следующие: неучет зависимостей между правилами; ошибочность, противоречивость и неполнота правил; неправильно выбрана модель представления знаний.

2.1. ПОДГОТОВКА И КОМПИЛИРОВАНИЕ ИСХОДНОГО ТЕКСТА ПРОГРАММЫ ЭС

Исходной информацией для написания текста программы ЭС на входном языке оболочки являются результаты формализации знаний или формализованная модель решения задачи. Она представляется в виде иерархии продукционных правил, описания входных и промежуточных переменных, перечня тестов и эталонов для проверки правильности и точности решения задачи.

Процесс подготовки (программирование, кодирование) исходного текста программы ЭС будем рассматривать как преобразование результатов формализации знаний в конструкции входного языка оболочки GURU.

Подготовка исходного текста программы. В связи с тем, что структура программы ЭС на языке оболочки имеет строго фиксированную структуру, то для подготовки текста программы ЭС рекомендуется использовать методику, включающую следующие действия:

1. ***Определение раздела цели*** предполагает описание целевой переменной ЭС. В качестве целевой переменной используется целевая переменная, которая определена на этапе структурирования знаний, а затем уточнена в процессе формализации. Процедура определения целевой переменной реализуется в рамках сценария, который рассмотрен в п.1.6.2. (см. рис.1.4.).
2. ***Определение раздела описания переменных*** включает описание рабочих (входные и промежуточные) переменных программы ЭС, которые определены на этапе формализации. Процедура определения этого раздела реализуется в рамках сценария, который рассмотрен в п. 1.6.6. (см. рис.1.8).
3. ***Определение раздела инициализации входных переменных*** предполагает описание последовательности команд, выполняемых до обращения к разделу правил. В этом разделе описываются все входные и промежуточные переменные, а также целевая переменная. Процедура определения этого раздела реализуется в рамках сценария, который рассмотрен в п. 1.6.3. (см. рис.1.5).
4. ***Определение раздела завершающих действий.*** В этом разделе определяются завершающие действия, которые выполняются по окончании выполнения ЭС. Этот раздел обычно включает нахождение значения целевой переменной ЭС. Процедура формирования данного раздела реализуется в рамках сценария, который рассмотрен в п. 1.6.4. (см. рис.1.6).
5. ***Определение раздела правил.*** В этом разделе определяются правила программы ЭС. Исходной информацией для формирования правил выступают результаты формализации модели решения задачи, которые представляются в виде иерархии продукционных правил. Процедура определения правил реализуется в рамках сценария, который рассмотрен в п. 1.6.5. (см. рис.1.7).

Проверка предполагает использование различных ручных способов проверки работы программы ЭС.

Компилирование текста программы ЭС на языке оболочки можно выполнять в диалоговом режиме путем выбора ***пункта Compile*** меню в ***блоке 4*** (см. рис.1.2). Выдача ошибок компиляции программы ЭС реализуется средствами оболочки. Поиск и исправление ошибок выполняется разработчиком программы путем использования средств редактирования исходного текста программы и повторного его компилирования.

2.2. ОТЛАДКА И ТЕСТИРОВАНИЕ ПРОГРАММЫ ЭС
2.2.1. ОБЩАЯ СХЕМА ОТЛАДКИ ПРОГРАММЫ ЭС

Отладка – это процесс, позволяющий получить программу ЭС, функционирующую с требующимися характеристиками в заданной области изменения входных данных. В результате отладки программа ЭС должна соответствовать некоторой фиксированной совокупности правил и показателей качества, принимаемых за эталон для данной программы.

Тестирование – это основной метод обнаружения ошибок при отладке программ. При этом затраты на тестирование для обнаружения ошибок являются наибольшими, достигают 30-40 % общих затрат на разработку программ и в значительной степени определяют качество созданной программы. Высокая доля затрат на тестирование приводит к необходимости создания методов и средств, позволяющих достигать максимального качества программ при реальных ограничениях на длительность тестирования и на связанные с этим затраты.

Эффективность тестирования является важнейшим фактором, определяющим стоимость и длительность разработки ЭС. При этом используются различные методы систематического и регламентированного тестирования, обеспечивающие наилучшее использование ресурсов с учетом особенностей создаваемых программ.

ЭС – это программное изделие, пригодное для использования на ЭВМ, прошедшее испытания с зафиксированными показателями качества и снабженное комплектом документации, достаточной для квалифицированной эксплуатации по назначению и использованию.

Общая схема процесса отладки программ изображена на рис.2.1. Организация и проведение процесса отладки программы ЭС включает выполнение последовательности проверок (тестов) по схеме, которая представлена на рис.2.1. Она включает следующие действия:

1. ***Разработка критериев и создание совокупности тестов и эталонов*** (блок 1 на рис.2.1), схема формирования которых приведена на рис.2.2;
2. ***Тестирование для обнаружения ошибок*** (блоки 2, 5 и 8 на рис.2.1) в программе ЭС. При этом возможно использование различных методов тестирования: статическое, динамическое, детерминированное, стохастическое и тестирование в реальном масштабе времени [11];
3. ***Тестирование для диагностики и локализации причин*** (блоки 3, 6 и 9 на рис.2.1), *вызвавших искажения* в программе ЭС;
4. ***Контрольное тестирование*** (блоки 4, 7 и 10 на рис.2.1) программы ЭС.

Для определения задач тестирования целесообразно выделить три стадии:

1. ***Тестирование для обнаружения ошибок*** является основной целью выявления всех отклонений результатов функционирования реальной программы от заданных эталонных значений. При этом задача состоит в обнаружении максимального числа ошибок, в качестве которых принимается любое отклонение от эталонов. На этой стадии успешным является тестирование, которое приводит к обнаружению ошибок. Если в результате тестирования ошибки не выявлены, то проведенные операции не дали сведений, которые позволяют повысить качество программ и тем самым не оправдали затрат. Эффективными являются операции тестирования, обладающие высокой способностью по обнаружению ошибок в программе. Чем больше ошибок выявляется на этой стадии при каждой операции тестирования, тем выше их эффективность и обоснованность затрат на их выполнение. С этих позиций тесты, не способствующие обнаружению ошибок и только подтверждающие корректность функционирования программ, являются неэффективными, так как приводят к бесполезным затратам.

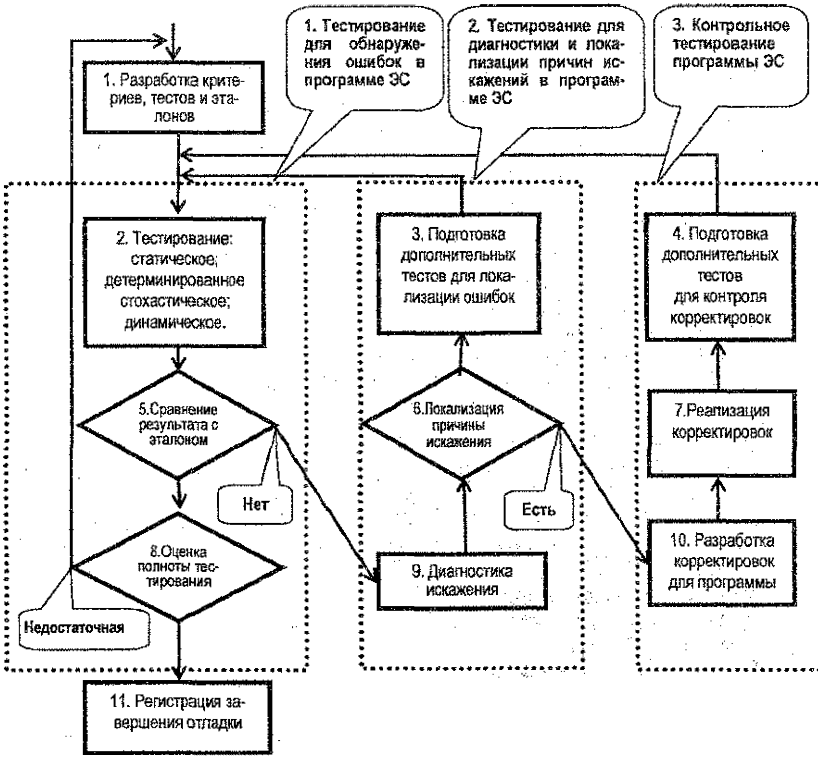


Рисунок 2.1 – Общая схема отладки программ

2. ***Тестирование диагностики и локализации*** используется после тестирования для обнаружения ошибок. Его цели и методы несколько отличаются от тех, которые соответствуют первой стадии тестирования. На этой стадии важнейшая задача – точно установить место искажения программы или данных, явившегося причиной отклонения результатов от эталонных при тестировании для обнаружения ошибок. Тем самым определяется часть программы, подлежащая корректировке. Эффективными являются тесты, способствующие быстрой и точной локализации первичных ошибок. На этой стадии затраты оправданы, и тестирование можно считать успешным, если оно приводит к полной локализации ошибки, подлежащей исправлению.

3. ***Контрольное тестирование*** применяется после локализации и устранения обнаруженных ошибок. Его задача состоит в подтверждении правильности выполненной корректировки программы и в отсутствии проявления ранее обнаруженных ошибок. В этом случае успешность тестирования определяется отсутствием проявления ранее обнаруженной, локализованной и устраненной ошибки, а также отсутствием вторичных ошибок, которые могут появиться при корректировке.

2.2.2. ОБЩИЕ СВЕДЕНИЯ О ТЕСТИРОВАНИИ ПРОГРАММНЫХ МОДУЛЕЙ

Программу ЭС можно рассматривать как программный модуль, тестирование которого предполагает выполнение следующей последовательности проверок [1,2]:

1. ***Проверка интерфейса модуля.*** Интерфейс модуля тестируется для проверки правильности ввода-вывода тестовой информации. Если нет уверенности в правильном вводе-выводе данных, нет смысла проводить другие тесты.
2. ***Проверка внутренних структур данных.*** Проверка внутренних структур данных гарантирует целостность сохраняемых данных.
3. ***Проверка структуры программного модуля*** предназначена для выявления ошибок в схеме принятия решений и логике функционирования модуля. Должна быть проверена корректность структуры модуля и их основные конструктивные компоненты: циклы, логические операторы и т.д. Проверке подлежат маршруты обработки информации в модуле и правильность их реализации в зависимости от исходных данных. Полнота проверки определяется критериями выделения маршрутов для тестирования и степенью покрытия маршрутов исполнения программы.
4. ***Проверка вычислений и преобразований данных программным модулем*** служит для обнаружения ошибок в вычислительной части программы. Для этого выделяются компоненты программного модуля и маршруты обработки данных, содержащие вычисления и преобразования переменных. В состав теста входят наборы значений исходных переменных во всей области их определения. Особо выделяются комбинации переменных в критических и особых точках, а также сочетания противоречивых и искаженных данных. Эталонами служат результаты предварительных расчетов по формулам, заложенным в модуле.
5. ***Проверка полноты функций,*** выполняемых модулем, предназначена для выявления ошибок в разработанной программе относительно функций, заданных в программной спецификации. Тестированию подлежат реализация всех функций, определенных спецификацией, и характеристики интерфейса. Для проверки функций используются результаты тестирования структуры и вычислений, которые дополняются тестами для проверки не контролировавшихся функций.

Оценку качества структуры программы можно проводить на основе показателя **корректность структуры программы** [2]. Для оценки этого показателя наиболее часто используются несколько частных показателей – критериев выделения маршрутов программы для тестирования – критерии К1, К2 или К3 [2], которые различаются степенью охвата тестами структуры программы и необходимыми ресурсами для их реализации.

Критерий выделения маршрутов программы для тестирования – это критерий, который определяет полноту покрытия графа программы маршрутами проверки. Например [2]:

- **Критерий К1** — покрытие графа программы минимальным количеством маршрутов, охватывающих каждую дугу хотя бы один раз (минимальное покрытие). Согласно критерию $к_1$ должны быть разработаны тесты, которые обеспечивают покрытие всех блоков решений в программе. При этом каждое решение должно принять значение «истина» и «ложь», а также должна исполниться каждая логическая ветвь программы.
- **Критерий К3** — покрытие графа программы маршрутами при всех возможных комбинациях дуг (максимальное или полное покрытие). Согласно критерию $к_2$ должны быть разработаны тесты, обеспечивающие выполнение всех возможных комбинаций результатов условий в каждом решении. Этот критерий является наиболее полным.

Критерии приведены в порядке углубления тестовых проверок, увеличения объема тестов и достигаемой степени корректности программ.

Проверку корректности структуры программы будем проводить статически (ручным способом) на основе схемы алгоритма программы модуля, которая задается в качестве исходных данных для лабораторной работы.

Граф программы. Для удобства рассмотрения схем алгоритмов программ используется их представление в виде графа программы. Для этого необходимо заменить блоки (операторы) схемы на вершины графа.

Маршрут проверки – это совокупность конкретных вершин и дуг на графе программы, которые выполняются в рамках данного маршрута. Некоторые маршруты могут оказаться нереализуемыми из-за несовместимости условий, которые последовательно анализируются в разных вершинах маршрута. Эти маршруты отбраковываются.

Тест – это отдельная проверка программы модуля, которая определяется следующими компонентами: маршрутом проверки; совокупностью значений входных данных модуля (**входной набор теста**), определяющих выполнение программы по данному маршруту; совокупностью значений выходных данных модуля (**эталон** для проверки – **выходной набор теста**), рассчитанных для данного маршрута на основе входного набора.

Упорядочение тестов. Для упорядочения разработанных тестов для проверки модуля можно использовать следующие критерии: по длительности исполнения или по числу команд в маршруте; по количеству условных операторов в маршруте; по вероятности исполнения маршрута и другие. По первому критерию первичной проверке подвергаются маршруты, наиболее длинные либо по числу команд, либо по времени исполнения. По второму критерию приоритет отдается маршрутам наиболее сложным по числу условий. По третьему критерию в первую очередь выполняются маршруты с наибольшей вероятностью их реализации.

2.2.3. МЕТОДИКА РАЗРАБОТКИ ТЕСТОВ ДЛЯ ПРОВЕРКИ СТРУКТУРЫ МОДУЛЯ

Методика разработки тестов для проверки структуры модуля включается последовательность следующих этапов:

1. ***Выбор критерия выделения маршрутов для проверки.*** Эти критерии определяют полноту проверки структуры модуля. Тестирование структуры программы предполагает разработку групп тестов в соответствии с выбранным критерием полноты проверки структуры программы, который позволяет в той или иной степени проверить прохождение различных маршрутов программы.
2. ***Разработка набора маршрутов.*** В соответствии с выбранным критерием разрабатывается соответствующее множество маршрутов для проверки структуры программы. Каждый маршрут определяется в виде последовательности проверяемых вершин и дуг графа программы.
3. ***Разработка тестов.*** Для каждого маршрута разрабатывается свой тест: подбирается конкретный набор значений входных данных для данного маршрута; рассчитывается соответствующий набор выходных данных (эталон), которые соответствуют заданному маршруту.
4. ***Анализ маршрутов и отбраковка нереализуемых маршрутов.*** Каждый маршрут анализируется на предмет его реализуемости. Для этой цели рассматриваются логические выражения в логических операторах анализируемого маршрута. В случае, если обнаруживаются несовместимые логические выражения в разных вершинах маршрута, например, X>2 и X<1 (X не может одновременно быть больше двух и меньше единицы), то такой маршрут является нереализуемым и выбрасывается из дальнейшего рассмотрения.

5. ***Выбор критерии для упорядочения тестов.*** Критерий (критерии) выбирается на основе анализа результатов разработки тестов и наличия необходимой информации для их применения. При отсутствии упорядоченного анализа структуры модуля некоторое число маршрутов в модуле может оказаться пропущенным при проверке.
6. ***Упорядочение тестов.*** Оно выполняется для тестов в соответствии с выбранным критерием упорядочения маршрутов.

Процесс разработки набора тестов для проверки структуры программы отдельного модуля рассмотрим для модуля, схема алгоритма которого представлена на рис.2.1. Обозначения блоков в результирующих таблицах (табл.2.1. и табл.2.2) приведены с использованием их представления на схеме алгоритма.

Этап 1. Выбор критериев выделения маршрутов для покрытия графа программы. Процесс разработки маршрутов рассмотрим для двух вариантов критериев – минимального и максимального покрытия графа программы, которые определены в п.2.1.

Этап 2. Разработка маршрутов. В соответствии с перечисленными выше критериями разработано два набора маршрутов проверки, которые представлены в табл.2.1. Первая группа маршрутов (критерий К1) включает 4 маршрута. Эти маршруты минимальным образом покрывают граф программы. Для этого критерия повторная проверка дуг не оценивается и считается избыточной.

Вторая группа представлена 12 маршрутами (критерий К2). В этой группе представлены все возможные комбинации дуг в маршрутах.

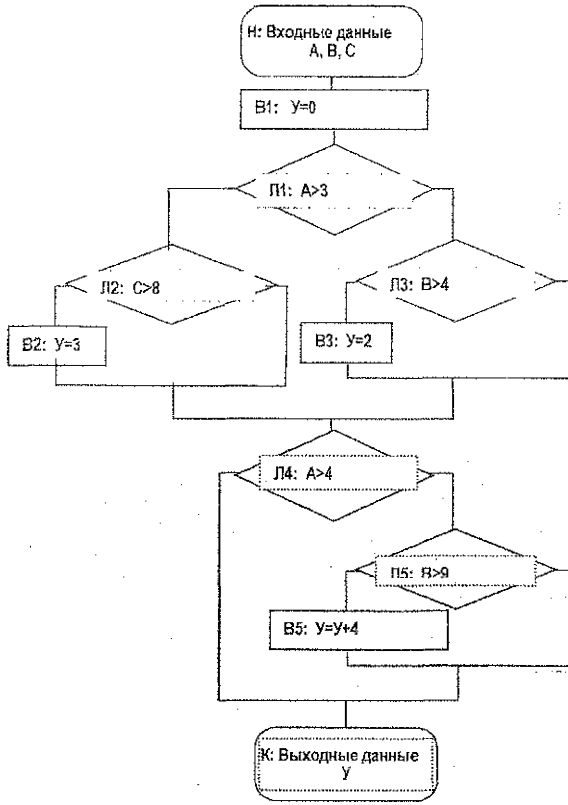
Для каждого маршрута из рассмотренных групп в таблице описания тестов приводится:

1. Определение элементов маршрута проверки в виде последовательности блоков (операторов) схемы алгоритма, которые вошли в маршрут. Рекомендуется его представлять в виде наименований блоков из схемы алгоритма, через которые он проходит. Например, В1-Л1-Л3-Л4-Л5-К и т.д. При этом использованы следующие обозначения:
   - ВN – символ В обозначает вычислительный блок, а N – порядковый номер этого блока;
   - ЛN – символ Л обозначает логический блок, а N – порядковый номер этого блока;
   - Н и К – соответственно начальный и конечный блоки схемы алгоритма.
2. Документирование логических условий по маршруту, которые используются в логических блоках. Эта информация используется для определения нереализуемых маршрутов.

Этап 3. Разработка тестов. Для каждого маршрута разрабатывается отдельный тест, который включает определение следующей информации:

1. Присвоение тесту идентификатора (каждому тесту присваивается порядковый номер).
2. Подбор набора значений для входных данных модуля, которые определяют реализацию данного маршрута;
3. Расчет значений выходных данных модуля (эталон) для данного набора входных данных.

Результат разработки тестов для двух групп маршрутов представлен в табл.2.1.



Примечание:

В качестве исходных данных модуля использованы переменные A, B, C. В качестве выходных данных определена переменная У. На рис.2.2 использованы следующие обозначения:

- **В1, ..., В5** – обозначения блоков (операторов), в которых выполняются расчеты. Для каждого отдельного блока представлены расчетные формулы для выполнения вычислений;
- **Л1, ..., Л5** – обозначения логических блоков (операторов), в которых выполняются логические операторы. Логическое выражение задается в виде логического выражения над входными данными.

Рисунок 2.2 – Схема алгоритма программы модуля

Этап 4. Анализ и отбраковка нереализуемых маршрутов. В первой группе тестов (критерий К1) нереализуемых маршрутов не выявлено. Для второго набора тестов было выявлено четыре маршрута (см.табл.2.1), 3, 4, 7 и 8 для которых выявлены – несовместимые логические условия A=<3 и A>4. Перечисленные маршруты отбракованы, т.е. выброшены из дальнейшего рассмотрения.

Этап 5. Выбор критерия для упорядочения тестов. Для упорядочения маршрутов использованы следующие критерии: упорядочение по количеству логических операторов и по длительности выполнения маршрута. Значения этих критериев (количество логических и вычислительных операторов для каждого маршрута) приведены в табл.2.2.

Этап 6. Упорядочение тестов. Результаты упорядочения разработанного набора тестов приведены в табл.2.2. Как представлено в табл.2.2, количество логических блоков для всех вариантов тестов равно четырем. Поэтому упорядочение тестов по этому параметру не имеет смысла. Упорядочение выполнено по второму критерию (количество вычислительных блоков) и представлено в графе «Последовательность выполнения тестов» (см. табл.2.2).

Таблица 2.1 – Описание тестов для проверки структуры модуля

Критерий покрытия графа программы	Входные данные для теста			Выходные данные (эталон)	Проверяемые условия на маршруте	Определение маршрута проверки	Номер теста
	A	B	C	У			
К1	5	10		0	A>3, B>4, A>4, B>9	Н-В1-Л1-Л3-Л4-Л5-К	1
	5	4		6	A>3, B=<4, A>4, B=<9	Н-В1-Л1-Л3-В3-Л4-Л5-В5-К	2
	3		9	0	A=<3,C>8, A=<4	Н-В1-Л1-Л2-Л4-К	3
	3		9	3	A=<3,C=<8, A=<4	Н-В1-Л1-Л2-В2-Л4-К	4
К2	5	5		0	A>3,B>4,A>4, B>9	Н-В1-Л1-Л3-Л4-Л5-К	1
	5	4		2	A>3,B=<4, A>4, B>9	Н-В1-Л1-Л3-В3- Л4-Л5-К	2
				0	**A=<3,C>8, A>4, B>9**	**Н-В1-Л1-Л2- Л4-Л5-К**	**3**
				0	**A=<3,C=<8, A>4, B>9**	**Н-В1-Л1-Л2-В2- Л4-Л5-К**	**4**
	2	5		4	A>3, B>4, A>4, B=<9	Н-В1-Л1-Л3-Л4-Л5-В5-К	5
	5	4		0	A>3, B=<4, A>4, B=<9	Н-В1-Л1-Л3-В3- Л4-Л5-В5-К	6
				4	**A=<3,C>8, A>4, B=<9**	**Н-В1-Л1-Л2- Л4-Л5-В5-К**	**7**
				6	**A=<3,C=<8, A>4, B=<9**	**Н-В1-Л1-Л2-В2- Л4-Л5-В5-К**	**8**
	4	5		0	A>3, B>4, A=<4	Н-В1-Л1-Л3-Л4-Л5-В5-К	9
	4	4	9	2	A>3, B=<4, A=<4	Н-В1-Л1-Л3-В3- Л4-Л5-В5-К	10
	2		9	0	A=<3,C>8, A=<4	Н-В1-Л1-Л3-Л4-Л5-В5-К	11
	3		8	3	A=<3,C=<8, A=<4	Н-В1-Л1-Л3-В3- Л4-Л5-В5-К	12

Таблица 2.2 – Результаты анализа и упорядочения тестов

Номер теста из табл.2.1	Маршрут проверки	Количество логических блоков в маршруте	Количество вычислительных блоков в маршруте	Последовательность выполнения тестов
1	Н-В1-Л1-Л3-Л4-Л5-К	4	1	8
2	Н-В1-Л1-Л3-В3- Л4-Л5-К	4	2	4
5	Н-В1-Л1-Л3-Л4-Л5-В5-К	4	2	5
6	Н-В1-Л1-Л3-В3- Л4-Л5-В5-К	4	3	1
9	Н-В1-Л1-Л3-Л4-Л5-В5-К	4	2	6
10	Н-В1-Л1-Л3-В3- Л4-Л5-В5-К	4	3	2
11	Н-В1-Л1-Л3-Л4-Л5-В5-К	4	2	7
12	Н-В1-Л1-Л3-В3- Л4-Л5-В5-К	4	3	3

2.2.4. ПРИМЕР ПРОЦЕДУРЫ РАЗРАБОТКИ КРИТЕРИЕВ, ТЕСТОВ И ЭТАЛОНОВ ДЛЯ ПРОВЕРКИ ПРОГРАММЫ ЭС

Общая схема процедуры определения критериев, разработки тестов и эталонов для организации отладки программы ЭС представлена на рис.2.3. Эта процедура включает последовательность следующих действий:

1. ***Определение критериев*** *для* оценки достижения полноты проверки программы ЭС и завершения процесса ее отладки. Например, критерий для полноты проверки структуры программы и/или критерий для проверки обработки данных [10];

2. ***Разработка тестов и эталонов*** в соответствии с определенными выше критериями для оценки;
3. ***Выбор критерия для упорядочения тестов и упорядочение тестов*** для определения последовательности их выполнения. Например, по времени реализации проверки или другие [10].

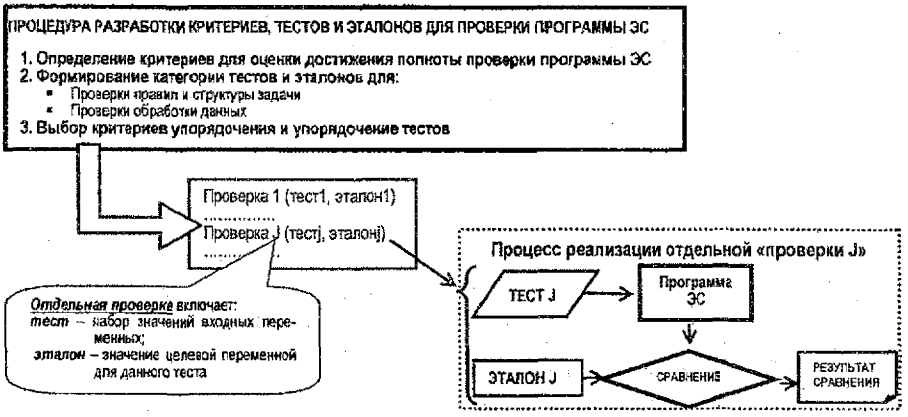


Рисунок 2.3 – Общая схема определения критериев, разработки тестов и эталонов для проверки программы ЭС

Для проверки программы учебной ЭС (см. рис.1.9) в таблице 2.3 приведено три теста (наборы значений входных параметров программы) и три эталона.

Таблица 2.3 – Пример тестов и эталонов для отладки программы ЭС

Имена переменных		Значение переменных для теста 1	Значение переменных для теста 2	Значение переменных для теста 3
Входные переменные	DEGREE	Нет	Да	Да
	DISCOVERY		Да	Нет
	GRADE			4
	EXPERIENCE			
Выходная (целевая) переменная		Значение эталона для теста 1	Значение эталона для теста 2	Значение эталона для теста 3
	POSITION	«Отказать»	«Научный Сотрудник»	«Инженер Конструктор»

2.3. РАЗРАБОТКА ДОКУМЕНТАЦИИ ДЛЯ ЭС

Документация на программу ЭС предназначена для детального отображения ее содержания и специфики в процессе проектирования, отладки, изготовления, эксплуатации и сопровождения.

Документация должна разрабатываться с самого начала создания ЭС. Значительные запаздывания в корректировке документов могут приводить к пропуску изменений и к появлению несоответствия документов работающей программе.

Для небольших ЭС (представляют собой программное изделие) процесс их создания и сопровождения определяется ГОСТами единой системы программной документации (ЕСПД) [13-21]. Для ЭС, которые относятся к классу автоматизированных систем, создание и сопровождение определяется ГОСТами для автоматизированных систем [12].

Основные положения ГОСТ ЕСПД. К программным документам (ПД) относятся документы, содержащие сведения, необходимые для разработки, изготовления, сопровождения и эксплуатации программ. В соответствии с ГОСТ ЕСПД к основным ПД относятся:

1. **Текст программы** – запись программы с необходимыми комментариями.
2. **Описание программы** – сведения о логической структуре и функционировании программы.
3. **Программа и методика испытаний** – требования, подлежащие проверке при испытании программы, а также порядок и методы их контроля.
4. **Техническое задание** (ТЗ) – назначение и область применения программы, технико-экономические и специальные требования, предъявляемые к программе, необходимые стадии и сроки разработки, виды испытаний.
5. **Эксплуатационные документы** (ЭД) – сведения для обеспечения функционирования и эксплуатации программы (описание применения, руководство программиста и другие).

Эксплуатационные документы разрабатываются и изготовляются в соответствии с ГОСТами ЕСПД, которые определяют структуру и содержание обязательных документов, минимально необходимых для эксплуатации ЭС, и вспомогательных документов, разрабатываемых по мере необходимости. Полный состав ЭД для конкретной ЭС зависит от класса ЭС и уточняется по согласованию с заказчиком.

Содержание документов должно обеспечивать эффективную реализацию технологии разработки и сопровождения программы ЭС. Ряд документов может оформляться вне имеющихся стандартов, однако их полнота и содержание должны обеспечивать возможность сопровождения программы и последовательной смены версий у пользователей. Правила комплектования, корректировки и хранения ЭД должны гарантировать однозначное соответствие ЭД конкретным версиям ЭС и содержанию программы.

Для небольших продукционных ЭС рекомендуется разрабатывать перечень следующих документов:

1. Постановка задачи на создание и сопровождение ЭС – документ «Техническое задание на создание ЭС». Пример фрагмента документа приведен в [11]. Структура документа – [12,17];
2. Документация на программу – документы «Текст программы» [19] и «Описание программы» [20];
3. Руководство по применению ЭС конечными пользователями – документ «Описание применения ЭС» [21];
4. Испытание ЭС – документ «Программа и методика испытания ЭС» [18].

2.3.1. ТЕКСТ ПРОГРАММЫ-ПРОТОТИПА ЭС

Требования к содержанию и оформлению программного документа "Текст программы" установлены в ГОСТ 19.401-78 [17] – для программных изделий, а для автоматизированных систем – [12].

Основная часть документа должна состоять из текстов одного или нескольких разделов, к которым даны наименования. Допускается вводить наименование также и для совокупности разделов. Каждый из этих разделов реализуется одним из типов символической записи, например: символическая запись на исходном языке.

В символическую запись разделов рекомендуется включать комментарии, которые могут отражать, например, функциональное назначение, структуру.

2.3.2. ОПИСАНИЕ ПРОГРАММЫ-ПРОТОТИПА ЭС

Состав и требования к содержанию программного документа "Описание программы" устанавливает ГОСТ 19.402-78 [20]. Описание программы должно содержать следующие разделы:

- **Общие сведения.** Указываются: обозначение и наименование программы; программное обеспечение, необходимое для функционирования программы; языки программирования, на которых написана программа.
- **Функциональное назначение.** Указываются классы решаемых задач и (или) назначение программы и сведения о функциональных ограничениях на применение.
- **Описание логической структуры.** Указываются: алгоритм программы; используемые методы; структура программы с описанием функций составных частей и связи между ними; связи программы с другими программами. Описание логической структуры программы выполняют с учетом текста программы на исходном языке.
- **Используемые технические средства.** Указываются типы ЭВМ и устройств, которые используются при работе программы.
- **Вызов и загрузка.** Указываются: способ вызова программы с соответствующего носителя данных; входные точки в программу. Допускается указывать адреса загрузки, сведения об использовании оперативной памяти, объем программы.
- **Входные данные.** Указываются: характер, организация и предварительная подготовка входных данных; формат, описание и способ кодирования входных данных.
- **Выходные данные.** Указываются: характер, организация и предварительная подготовка выходных данных; формат, описание и способ кодирования выходных данных.

Допускается содержание разделов иллюстрировать пояснительными примерами, таблицами, схемами, графиками.

В приложение к описанию программы допускается включать различные материалы, которые нецелесообразно включать в разделы описания.

2.3.3. ОПИСАНИЕ ПРИМЕНЕНИЯ ПРОТОТИПА ЭС

Требования к содержанию программного документа "Описание применения" определено в ГОСТ 19.101-77 [21]. Текст документа должен состоять из следующих разделов:

- **Назначение программы.** Указываются назначение, возможности программы, ее основные характеристики, ограничения, накладываемые на область применения программы.
- **Условия применения.** Указываются условия, необходимые для выполнения программы (требования к необходимым для данной программы техническим средствам и другим программам, общие характеристики входной и выходной информации, а также требования и условия организационного и технического характера и т.п.).
- **Описание задачи.** Определяются задачи и методы ее решения.
- **Входные и выходные данные.** Указываются сведения о входных и выходных данных.

В приложение к общему описанию могут быть включены справочные материалы (иллюстрации, таблицы, графики, примеры и т.п.).

2.4. ИСПЫТАНИЕ ПРОТОТИПА ЭС

Испытание программы – это процесс доказательства соответствия созданной программы и программной документации требованиям, которые зафиксированы в документе «Техническое задание на разработку ЭС». Испытание программы осуществляется в соответствии с программным документом "Программа и методика испытаний". Требования к содержанию и оформлению этого документа определены в ГОСТ 19.101-2000 [18]. Данный документ разрабатывается после завершения комплексного тестирования программы и разработки указанной в ТЗ перечня ЭД.

Документ **"Программа и методика испытаний"** должен содержать следующие разделы:

- **Объект испытаний.** Указываются наименование, область применения и обозначение испытуемой программы.
- **Цель испытаний.** Указывается цель проведения испытаний.
- **Требования к программе.** Указываются требования, подлежащие проверке во время испытаний и заданные в ТЗ на программу. К этим проверкам относятся требования устойчивости функционирования программы при наличии ошибок во входных данных, возможности обработки ошибочных ситуаций, полноты обработки ошибочных ситуаций, по восстановлению процесса выполнения программы и другие;
- **Требования к программной документации.** Указывается: состав программной документации, предъявляемой на испытания (этот перечень определяется в ТЗ на разработку); перечень требований к полноте и понятности изложения в документации информации об изделии, его компонентах и т.д.
- **Средства и порядок испытаний.** Указываются технические и программные средства, используемые во время испытаний, а также порядок проведения испытаний. Порядок проведения испытаний предполагает определение последовательности выполнения процедур проверки корректности: функционирования программы на соответствие предъявленным требованиям; реализации всех основных функций; реализации всех частных функций.
- **Методы испытаний.** Приводятся описания используемых методов испытаний. Методы испытаний рекомендуется по отдельным показателям располагать в последовательности, в которой эти показатели расположены в разделах "Требования к программе" и "Требования к программной документации". В методах испытаний должны быть приведены описания проверок с указанием результатов проведения испытаний (перечней тестовых примеров, контрольных распечаток тестовых примеров и т.п.).

В приложение к документу могут быть включены тестовые примеры, контрольные распечатки тестовых примеров, таблицы, графики и т.п.

3. ЭКСПЛУАТАЦИЯ И СОПРОВОЖДЕНИЕ ЭС

Внедрение программы предполагает испытание и адаптацию программы для конкретной среды эксплуатации и сопровождения.

Эксплуатация программы – это процесс использования программы в соответствии с ее назначением, которое определено в эксплуатационных документах на программу.

Сопровождение программы – это процесс исправления возникающих при эксплуатации ошибок, расширение возможностей программы. Этот процесс ориентирован на обеспечение контроля и повышение качества, а также развитие функциональных возможностей ЭС.

ЭС являются одним из наиболее гибких видов программных изделий, которые эпизодически подвергаются изменениям в течение всего времени их использования. Иногда достаточно при корректировке внести хотя бы одну ошибку для того, чтобы резко снизить надежность программы ЭС или ее корректность при некоторых исходных данных.

Для сохранения и повышения качества ЭС необходимо регламентировать процесс модификации и поддерживать его соответствующим тестированием и контролем качества. В результате ЭС со временем обычно улучшаются как по функциональным возможностям, так и по качеству решения отдельных задач.

В процессе сопровождения в программы вносятся следующие виды изменений:

- **исправление ошибок в ЭС** – корректировка программ, выдающих неправильные результаты в условиях, ограниченных техническим заданием и документацией;
- **адаптация ЭС к условиям конкретного использования**, регламентированная документами и обусловленная характеристиками внешней среды или конфигурацией аппаратуры, на которой предстоит функционировать программам;
- **модернизация ЭС** – расширение функциональных возможностей или улучшение характеристик решения отдельных задач в соответствии с новым или дополненным техническим заданием.

Первый вид изменений является непредсказуемым и его трудно регламентировать. Остальные виды корректировок носят упорядоченный характер и проводятся в соответствии с заранее подготовленными планами и документами. Поэтому изменения, обусловленные ошибками, в большинстве случаев целесообразно по возможности накапливать и реализовывать, приурочивая к изменениям, регламентированным модернизациями. Однако некоторые ошибки могут приводить к необходимости срочного исправления программ. В этих случаях допустимо некоторое отставание корректировки документации при более срочном и регистрируемом исправлении самих программ.

Со временем, иногда через десятки лет, сопровождение ЭС прекращается. Это может быть обусловлено:

- разработкой более совершенных ЭС;
- прекращением использования сопровождаемой ЭС;
- нерентабельным использованием сопровождаемой ЭС.

Для того, чтобы со временем прийти к обоснованному решению о прекращении сопровождения ЭС, необходимо периодически оценивать эффективность эксплуатации и возможный ущерб от отмены сопровождения.

На стадии «Сопровождение» выполняются следующие виды работ:

- консультирование и обучение пользователей ЭС;
- сбор и исправление ошибок в программе ЭС;
- расширение и модернизация существующих возможностей ЭС;
- выпуск новых версий ЭС.

В процессе сопровождения рекомендуется вести следующие документы:

- журнал ошибок, зафиксированных пользователями;

- журнал пожеланий, замечаний и рекомендаций по совершенствованию и модификации ЭС;
- журнал исправлений ошибок в ЭС разработчиком;
- каталог версий ЭС;
- документация на ЭС, отредактированная с учетом изменений, вызванных исправлением ошибок.

СПИСОК СОКРАЩЕНИЙ

БЗ - База знаний
ГОСТ - Государственный стандарт
ЕСПД - Единая система программной документации
МЛВ - Механизм логического вывода (машина вывода)
МРЗ - Модель решения задачи
ПД - Программный документ
ПрО - Предметная область
ТД - Технологическая документация
ЭД - Эксплуатационный документ
ЭО - Экспертная оболочка
ЭС - Экспертная система
ЭТ - Электронная таблица

СПИСОК ЛИТЕРАТУРЫ

1. Искусственный интеллект / Под ред. Д.А.Поспелова. – В 3-х кн. – М.: Радио и вязь, 1990.
2. Уотермен, У. Руководство по экспертным система / У. Уотерман; пер. с англ. – М.: Мир, 1989, – 388 с.
3. Элти, Дж. Экспертные системы: Концепции и примеры / Дж. Элти, М. Кумбис; пер.с англ. – М.: Финансы и статистика, 1986. – 191 с.
4. Экспертные системы. Принципы работы и примеры; пер. с англ. / Под ред. Р.Форсайта. – М.: Радио и связь, 1987. – 224 с.
5. Гаврилова, Т.А. Базы знаний интеллектуальных систем / Т.А. Гаврилова, В.Ф.Хорошевский – СПб.: Питер, 2001. – 384 с.
6. Голенда, Л.К. Экспертные системы и системы поддержки принятия решений: учебное пособие / Л.К. Голенда., М.А. Чепноков. – Мн.: БГЭУ, 1999. – 24 с.
7. Одинцов, Б.Е. Проектирование экономических экспертных систем: учебное пособие для вузов / Б.Е.Одинцов. – М.: Компьютер, ЮНИТИ, 1996. – 166 с.
8. Экспертные системы для персональных компьютеров: методы, средства, реализации: справочное пособие. /В.С.Крисевич, Л.А.Кузьмич, А.М.Шиф [и др.] – Мн.: Вышейшая школа, 1990. – 190 с.
9. Смородинский, С.С. Алгоритмы и программные средства интеллектуальных систем принятия решений: учебное пособие по курсу «Интеллектуальные системы принятия решений» / С.С. Смородинский, Н.В.Батин. – Минск, МРТИ, 1993. – Ч.2.– 78 с.
10. Липаев, В.В. Проектирование программных средств: учебное пособие для вузов по спец. «Автоматизированные системы обработки информации и управления» /В.В. Липаев – М.: Высш.шк., 1990. – 303 с.
11. Хвещук, В.И. Методические указания по дисциплине «Технология и инструментальные средства проектирования интеллектуальных систем. Проектирование экспертных систем». /В.И. Хвещук – Брест: БрГТУ, 2008г. – 52 с.
12. Информационная технология. Комплекс стандартов и руководящих документов на автоматизированные системы. – М.: Изд-во стандартов, 1991. – 144 с.
13. Виды программ и программных документов. ГОСТ ЕСПД 19.101-77.
14. Стадии разработки программ. ГОСТ ЕСПД 19 102-77.
15. Обозначение программ и программных документов. ГОСТ ЕСПД 19.103-77.
16. Общие требования к программным документам. ГОСТ ЕСПД 19.105-78.
17. Техническое задание. ГОСТ ЕСПД 19.201-78.
18. Программа и методика испытаний. ГОСТ ЕСПД 19.301-2000.
19. Текст программы. ГОСТ ЕСПД 19.401-78.
20. Описание программы. ГОСТ ЕСПД 19 402-78.
21. Описание применения. ГОСТ ЕСПД 19.502-78.

ПРИЛОЖЕНИЕ 1. ОСНОВНЫЕ ПЕРЕМЕННЫЕ СРЕДЫ ОБОЛОЧКИ

В данном приложении приведены основные переменные оболочки GURU. Для каждой переменной представлено назначение, описание типа и значение по умолчанию.

Таблица П.1.1 – Основные переменные среды

Переменная	Назначение	Тип	По умолчанию
E.BELL	Звуковой сигнал при вводе недопустимых данных	Логический	TRUE
E.BEST	Максимальное количество лучших(имеющих максимальный фактор уверенности) значений нечеткой переменной, используемых при оценке выражений, содержащих эту переменную	Числовой	4
E.CFCO	Способ объединения факторов уверенности для посылки, содержащей связку OR: M (максимум), P (вероятностная сумма), A (среднее вероятностной суммы и максимума).	Строка	M
E.CFCJ	Способ объединения факторов уверенности для посылки, содержащей связку AND: M (минимум), P (вероятностное произведение), A (среднее вероятностного произведения и минимума).	Строка	M
E.CFVA	Первая буква – способ объединения факторов уверенности посылки и заключения; значение и способ объединения – см. E.CFJO, вторая буква – способ объединения факторов уверенности переменной, полученной в нескольких правилах; значение и способ объединения см. E.CFCO	Строка	PP
E.DECI	Количество цифр справа от десятичной точки в командах ввода/вывода	Числовой	2
E.GUID	Вывод на экран интерфейса GURU (системы меню)	Логический	TRUE
E.HELP	Автоматическая выдача подсказки при ошибке (если есть файл GGUIDE.HLP)	Логический	FALSE
E.HRES	Степень ответа на команду HOW: от 0 (нет ответа) до 6 (наиболее полный ответ)	Числовой	4
E.ICAS	Игнорировать различия верхнего и нижнего регистров при работе со строками	Логический	TRUE
E.ICF	Ввод в команде INPUT фактора уверенности вместе со значением переменной	Логический	FALSE
E.IFUZ	Количество автоматических повторений команды INPUT для ввода значений нечеткой переменной	Числовой	1
E.LLOG	Длина шаблона для ввода логических переменных	Числовой	5
E.LNUM	Длина шаблона для ввода числовых данных (не более14)	Числовой	14
E.LSTR	Длина шаблона для ввода строковых данных	Числовой	15
E.NUMV	Максимальное количество значений, которое может иметь нечеткая переменная (не более 255)	Числовой	4
E.OCF	Вывод в команде OUTPUT фактора уверенности вместе со значением переменной	Логический	FALSE
E.OFUZ	Количество автоматических повторений команды OUTPUT для вывода значений нечетких переменных	Числовой	1
E.RIGR	Степень точности при выводе значений нечеткой переменной: A – использовать все возможные правила; M – минимальный вывод (прекращается, когда выводится значение переменной с фактором уверенности, превышающим порог E.UNKN); C – тщательный вывод (после достижения порога уверенности проверяется также все правила, для оценки которых не требуется выполнять вывод)	Строка	M

Переменная	Назначение	Тип	По умолчанию
E.SORD	Порядок рассмотрения конкурирующих правил: F – в порядке расположения в наборе правил, P – по приоритету, C – по минимальной стоимости, H – по максимальному фактору уверенности для текущей цели, U – по минимальному количеству неизвестных переменных в посылке, R случайном порядке	Строка	F
E.TRAC	Тип трассировки во время консультации с ЭС: N – трассировка не выполняется, C – выводятся рассмотренные и включенные правила, V – все рассматриваемые правила, текущая цель, включенные правила	Строка	N
E.TRYP	Стратегия оценки посылки, содержащей неизвестные переменные: S – строгая (правило не включается, если не удается определить хотя бы одну переменную), P – терпеливая (проверка посылки по окончании поиска всех неизвестных переменных), E – эталонная (проверка посылки по окончании поиска каждой из неизвестных переменных)	Строка	S
E.UNKN	Пороговое значение фактора уверенности. Переменные с фактором уверенности ниже порога считаются неизвестными	Числовой	20
E.WHN	Код времени выполнения последовательности поиска переменных: F – сначала выполняется последовательность поиска, затем, если требуется – поиск в правилах; L – сначала поиск в правилах, затем – последовательность поиска; N – последовательность поиска никогда не выполняется	Строка	L

ПРИЛОЖЕНИЕ 2. ОСНОВНЫЕ ОПЕРАЦИИ И ФУНКЦИИ

В данном приложении перечислены основные операции и функции оболочки GURU.

Арифметические операции:

+ – сложение;
- – вычитание;
* – умножение;
/ – деление;
** – возведение в степень;
MOD – деление по модулю.

Операции сравнения:

EQ, = – равно;
NE, <> – не равно;
GT, > – больше чем;
LT, < – меньше чем;
GE, <= – меньше или равно;
LE, >= – больше или равно.

Логические операции:

NOT – нет;
AND, & – и;
OR – или;
XOR – исключающее "или";
= – присвоение;
( ) – индексы массива.

Строковые операции:

+ – сцепление строк;
' – кавычка;
$ – символ соответствия символа;
* – символ соответствия строки.

В оболочке GURU в выражениях могут применяться следующие функции:

Числовые функций:

ABS – абсолютное значение;
ARCSIN – арксинус;
EXP – e в степени;
INIT – инициализирует массив;
LEN – определяет длину строки;
LN – вычисляет натуральный логарифм;
LOG – вычисляет логарифм с основанием 10;
MAX – вычисляет наибольшее из двух чисел;
MENU – создает меню;
MIN – вычисляет меньшее из двух чисел;
RAND – случайное число;
SIN – синус;
SQRT – квадратный корень.

Символьные функции:

CHR – преобразует код ASCII в его символьный эквивалент;
VAL – преобразует символ в его код ASCII;
INIT – инициализирует массив;
SUBSTR – выделяет подстроку из строки;
TIME – возвращает текущее время;
TOSTR – преобразует числа в символы;
TONUM – преобразует строку в число;
TRIM – отсекает конечные пробелы;
TYPE – тип переменной.

Логические функции:

ALPHASTR – вся ли строка состоит из букв;
INIT – инициализирует массив.

ПРИЛОЖЕНИЕ 3. КОМАНДЫ ОБРАБОТКИ ЭЛЕКТРОННЫХ ТАБЛИЦ

В данном приложении приведен перечень и форматы команд для обработки электронных таблиц оболочки GURU. В состав команд по обработке электронных таблиц относятся следующие команды:

\EDIT – вывод определенной текущей ячейки, которая в дальнейшем может быть отредактирована или изменена с помощью клавиш управления курсором. Установите курсор на ячейке, которую необходимо редактировать. После команды \EDIT можно изменить определение этой ячейки.

\UNDEFINE – удаление существующей определенной ячейки, включающее значение, тип представления и шаблон ячейки.

\UNDEFINE <ячейка> или **\UNDEFINE <блок ячеек>**, где <ячейка> – имя ячейки, <блок ячеек> – блок ячеек. Специфицируются двумя ячейками, которые указывают верхний левый и нижний правый углы блока ячеек. Если ячейка (или блок ячеек) опущена, то удаляется определение текущей ячейки. Примеры:

\UNDEFINE #A4 – удалить определение ячейки #A4
\UNDEFINE – удалить определение текущей ячейки.
\UNDEFINE #D8 #L25 – удалить определение блока ячеек с #D8 по #L25.

\USING – задать шаблон ячейки или блока ячеек.

\USING "шаблон" < ячейка >
\USING "шаблон" < блок ячеек >

"шаблон" – специфицирует редактирование, которое должна осуществлять "GURU";
<ячейка> или <блок ячеек> – см. описание команды ЭТ **"UNDEFINE"**.

Примеры :

\USING "(ddd)ddd-ddd' #C5
\USING "dd-dd" #A4 #G8

Все данные, которые будут вводиться в эту ячейку, будут преобразованы по шаблону.

\WIDTH – задать ширину столбцов. По умолчанию ширина столбцов равна 9.
\ **WIDTH <ширина столбца >.**
Минимальная ширина столбца – 5, максимальная – 255.
Примеры:
WIDTH #E20 – установить ширину столбца #E равной 20.
\COMPUTE – вычислитьт все значения ячеек ЭВ, основанные на текущих определениях.
\COMPUTE или **\COMPUTE < блок ячеек >**
Команда, введенная без параметров, вычисляет значения всех ячеек ЭТ.
Примеры: **\COMPUTE #D12 #M3**
\LET – присвоить переменной значение.
\LET< переменная >=< выражение >USING"шаблон".
Слово "LET" можно опустить. Примеры:
\LET #C5=9
\LET #A16=#D18+#D20
\#F12=SNUM USING "DD-DD-DDDD".
\COPY – копировать определение одной ячейки в другую.
\COPY < исходная ячейка > , < целевая ячейка >.
"GURU" высвечивает на экране: "Выравнивать (Y/N)?" Если Y, то
"GURU" устанавливает выражение относительно целевой ячейки.
Если N, то переменная указанной ячейки остается неизменной в определении целевой ячейки. Пример:
\COPY #A20, #C30 – копирует определение ячейки #A20 в ячейку #C30.
\SAVE – сохранить ЭТ в файле.
\ SAVE TO"файл"
Пример:
\SAVE TO TABLE – сохранить ЭТв файле TABLE.
\LOAD – загрузить ЭТ из файла.
\SAVE FROM "файл".
Пример:
\SAVE FROM TABLE – загрузить ЭТ из файла TABLE.
\STOP – закрыть ЭТ и возвращается в командный режим "GURU".
Пример:
\STOP.
\DISPLAY – выводит на печать необходимый участок или все значения ячеек ЭТ.
\DISPLAY <блок ячеек>
Печатный вывод имеет вид экрана, заканчивающегося границами. Убедитесь, что ваше печатающее устройство подсоединено и включено.
\DUMP – печать всей информации относительно заданных ячеек, которая включает:
имя ячейки, текущее значение, определение (если существует) и шаблон (если существует).
\ DUMP < блок ячеек >. Убедитесь, что печатающее устройство подсоединено и включено.

ПРИЛОЖЕНИЕ 4. ГРАФИЧЕСКИЕ СРЕДСТВА GURU

В данном приложении приведен перечень и форматы команд для обеспечения графических возможностей оболочки GURU. В состав этих команд входят следующие команды:

PLOT BAR – начертить гистограмму по исходным данным.
PLOT <тип гистограммы> BAR FROM <блок> AT <размещение><блок> – блок ячеек ЭТ или блок массивов. Если <блок> опущен, то подразумевается блок, специфицируемый в предыдущей команде PLOT.
<тип гистограммы> может принимать значение:
SOLID – трехмерная;
COMULATIVE – комулятивная;
STAKED – этажерочная (столбики в одной группе ставятся друг на друга).
<размещение> может принимать значение:
TOP, TOP LEFT, BOTTOM, TOP RIGHT, LEFT, BOTTOM LEFT, RIGHT, BOTTOM RIGHT.
Если AT <размещение> опущено, то граф занимает весь экран.

Примеры:

PLOT BAR FROM #E12 TO #G15

PLOT SOLID BAR FROM AR(1,1) TO AR(4,3) AT TOP PLOT STAKED BAR AT LEFT.

PLOT PIE – начертить круговую диаграмму ("пирог"). Приведем один из вариантов написания команды:

PLOT LABELED PERCENT PIE FROM. Команда должна сопровождаться параметрами: <источник данных> **AT**<размещение> <источник данных> – численное выражение, блок ячеек или блок массивов ЭТ. Если "**FROM** <источник данных>" опущено, то используется <источник данных> предыдущей команды PLOT.

Слово **PERCENT** в команде – необязательное. Если оно присутствует, то рядом с сегментами круговой диаграммы воспроизводятся проценты.

Слово **LABELED** может использоваться только, если все блоки в источнике данных имеют точно две колонки. Когда мы строим круговую диаграмму с использованием LABELED, значения отсчетов во второй колонке каждого блока используются для указания размеров сегментов. Эти сегменты маркированы соответствующими значениями из первых колонок (до 6 символов в каждом обозначении). AT < размещение> используется аналогично команде PLOT BAR. Примеры:

PLOT PIE FROM #L19 TO #N30 AT BOTTOM

PLOT LABELED PIE FROM #B5 TO #C15

PLOT PERCENT LABELED PIE FROM #A1 TO #B12

PLOT PIE FROM B(1,3) TO (2,7) AT BOTTOM.

PLOT LINE – начертить линейный граф.

PLOT LINE FROM < блок > AT < размещение>

<блок> – см. описание команды PLOT BAR; AT <размещение> – см. описание команды PLOT BAR. Линейный граф имеет X(горизонтальную) и Y(вертикальную)оси. Отдельные точки соединяются в графе линиями.

Примеры:

PLOT LINE FROM #G7 TO#MI2 AT LEFT

PLOT LINE FROM ARY(2,3) TO ARY(10,10);

PLOT FUNCTION – начертить функциональную линию.

PLOT FUNCTION = <выражение> FROM <нижняя граница> TO <верхняя граница> BY <приращение> AT <размещение> <выражение> – функциональная зависимость, которая будет выводиться на экран. <нижняя граница>,<верхняя граница> – область определения выводимого графика; <нижняя граница> должна быть меньше <верхней границы>.

Если <выражение> содержит переменную X, тогда X служит в качестве независимой переменной функции. Во время обработки этой команды "GURU" игнорирует любые ранее существовавшие переменные, которые имели имя X BY <приращение> – определяются дополнительные точки X-Y координат. AT <размещение> – см. описание команды PLOT BAR.

Примеры:

PLOT FUNCTION =X*X*4+1 FROM 5 TO 2 BY 1

PLOT FUNCTION =3/SQRT(X) FROM 15.2 TO 18.9 BY 3.5

PLOT FUNCTION =5*X*X*X BY.5

PLOT FUNCTION=#D5+X FROM 5 TO 50 BY 10 AT TOP.

RANGE – задает диапазон значений осей координат, используемых при построении графиков и контролирует расстояние между отметками на осях.

RANGE ACROS FROM <нижняя граница> TO <верхняя граница> BY<приращение>

<нижняя граница> <верхняя граница> <приращение> – см. описание предыдущей команды. Для графа, включающего числовые данные на осях X,Y, появляются ровно размещенные и помеченные отметки. Каждая ось может контролироваться независимо от других. Для оси X необходимо использовать ACROSS, для оси Y-UP.

PATTERN – задает тип заполнения, строку, отметку и типы цветов, используемых при вычерчивании графа.

PATTERN ORDER FOR AREA (LINE, MARK, COLOR) < коды >.

<коды > – список одного или нескольких численных выражений, указывающих порядок, в котором используются образцы командой PLOT. Например, если вводится команда PATTERN ORDER FOR AREA, то < коды > определяют тип заполнения площадей:

1 – вертикальные линии;

2 – горизонтальные, линии;

3,4,6 – диагональные линии;

5 – вертикальные + горизонтальные;

7 – непрерывный.

Если вводится PATTERN ORDER FOR LINE, то определяются линии на экране:

1 – линия ————————;
2,3,4 – линия — — —;
5 – линия ……;
6,7,8 – линия –.–.–.–;

Если вводится PATTERN ORDER COLOR, то коды определяют цвет для рисования линий и закрашивания площадей: R – красный; O -желтый; C – голубой; G – зеленый; U – синий; H – фиолетовый; W – белый; A – черный.

Например:

PATTERN ORDER FOR AREA 1,4,7,8

Тогда первая команда PLOT BAR выводит гистограмму с типом заполнения 1, вторая – 4, третья – 7, четвертая – 8. PATTERN ORDER FOR LINE 1

Все линии выводятся типа 1.

PATTERN ORDER FOR COLOR "RGOGRO"

Первый график будет цвета R,второй G,третий 0 и т.д.

PLOT TO – запись графического экрана в файл. PLOT TO < имя файла >

Например: PLOT TO "MY_FILE" PLOT TO "MY_GRAPH".

PLOT FROM – чтение графического экрана с файла. PLOT FROM < имя файла >. Например:

PLOT FROM "MY_FILE"

PLOT FROM "MY_GRAPH".

Пример программы, выводящей данные из ЭТ. Пусть имеется ЭТ, в которой определяется заработная плата:

	A	B	C
1	Месяц	Номер месяца	Размер оклада
2	Январь	1	1000
3	Февраль	2	1000
4	Март	3	1100
5	Апрель	4	1200
6	Май	5	1300
7	Июнь	6	1400

Напишем программу, которая выводит данные из этой ЭТ на экран в виде графов.

```
e.lstr=80 /*инициализация переменных*/
e.deci=2
e.lnum=8
while true do clear /*очистка экрана*/
vibor=0
output
output "выберите необходимый пункт"
output
output "1-Построение гистограммы"
output "2-Построение  круговой диаграммы"
output "3-Построение графика"
output "4-выход"
input vibor with "ваш выбор:"
test vibor
 case 1 :
  #title="Гистограмма"
  plot bar from #B2 to #C7
  break;
 case 2 :
  #title="Круговая диаграмма"
  plot labeled percent pie from #B2 to #C7
  break;
 case 3 :
  #title="Линейный граф"
  plot line from #B2 to #C7
  break;
 case 4 :
clear
return
  endtest
endwhile
```

ПРИЛОЖЕНИЕ 5. КОМАНДНЫЙ РЕЖИМ ВЗАИМОДЕЙСТВИЯ С GURU

В данном приложении приведен перечень и форматы команд для использования в командном режиме взаимодействия пользователя с оболочкой GURU. В состав этих команд входят следующие команды:

BUILD – создать набор правил из командной строки. Формат команды:

BUILD < имя – набора правил >.

После этого на экране появляется основное меню администратора набора правил системы "GURU". Пример:

BUILD MYEXPERT С помощью этой команды создается набор правил MYEXPERT и открывается исходный файл набора правил, называемый MYEXPERT.RSS.

COMPILE создает (транспирует) на основе текста набора правил ЭС, которую можно выполнить. Формат команды:

COMPILE < имя набора правил >

Пример:

COMPILE MYEXPERT В этом случае компилируется исходный файл набора правил MYEXPERT.RSS, который создает файл выполнимой ЭС MYEXPERT.RSC. Все ошибки и предупреждения появляются на экране и заносятся в файл MYEXPERT.WRN.

CONSULT – запускает программу ЭС для консультации. Формат команды:

CONSULT < имя экспертной системы >

Более расширенное использование команды CONSULT:

CONSULT [набор правил] [метод [объект]] [метод [объект]] определяет для машины логического вывода метод обработки;

[объект] – переменная цели, правило, список правил.

Возможны следующие четыре метода:

TO SEEK – выполняет обработку, используя для определения цели метод обратного сцепления;

TO TEST – выполняет обработку, используя для определения цели метод прямого сцепления;

TO FIRE – выполняет обработку, используя для запуска данного правила метод обратного сцепления;

TO EXECUTE – выполняет обработку списка данных правил, используя метод прямого сцепления.

Различные методы предполагают использование различных объектов. Для методов **"SEEK"** и **"TEST"** объект – это имя переменной цели. Если эта переменная не указана, тогда используется переменная цели, содержащаяся в наборе правил; для метода **"FIRE"** объект – это имя отдельного правила "GURU", которое "GURU" попытается запустить. В случае необходимости используется аргументация по методу обратного сцепления. По умолчанию в качестве объекта берется первое правило в наборе. Для метода **"EXECUTE"** объект – это список правил. Эти правила запускаются в данном порядке с применением метода прямого сцепления, если не учитывать того, что выполняется только один прогон. Примеры использования команды CONSULT:

CONSULT MYEXPERT TO TEST MYGOAL – для нахождения цели MYGOAL используется метод прямого сцепления.

CONSULT MYEXPERT TO FIRE 1 делается попытка запустить правило RULE1.

RUN – выполняет внешние программы (*.EXE и *.COM), работающие под управлением MSDOS. Система "GURU" продолжает прерванную работу после того, как внешняя программа выполнена. Примеры:

RUN COPY A: MYFILE.TXT C: – копирует файл MYFILE.TXT с диска A на диск C;

RUN PROG.EXE – запускает программу PROG.

DIR – вывод списка существующих файлов в директории. Пример: с помощью команды

DIR C:\MYDIR осуществляется просмотр файлов в директории MYDIR на диске C.

LET – присваивает значение переменным. Формат команды:

LET < переменная > = < выражение >.

Здесь: <переменная> – рабочая переменная, переменная поля или переменная среды; <выражение> – константа, переменная и операторы, которые выражают символьное, числовое или логическое значение. Примечание: слово LET необязательно. Примеры:

LET MYPER1 = "ABCD"

(аналогично: MYPER1 = "ABCD")

Присваивает переменной MYPER1 значение "ABCD".

MUPER2 = MUPER1

Присваивает переменной MUPER2 значение переменной MUPER1.

INPUT – ввод данных в переменную и имеет формат:

INPUT < переменная > [тип] [USING < шаблон >] [WITH < подсказка >].

где: <переменная> – см. комментарий к команде OUTPUT;

тип – см. комментарий к команде OUTPUT;

using < шаблон > – может принимать следующие значения:

a – для буквенных символов (латинский шрифт);

c – для буквы или числа;

d – для цифры, знака (+ или -) или десятичной точки;

e – преобразование в символы нижнего регистра;

n – символ шаблона, который воспринимает только цифры в занимаемой им позиций;

r – для символов ASCII;

u – преобразует в символы верхнего регистра.

Например:

INPUT num USING "dddd" with "Введите номер"

На экране появляется текст:

Введите номер __

Рассмотрим еще одну команду "GURU".

INPUT MYPER1 – вводится с экрана значение в MYPER1.

INPUT MYPER1 USING "aaa" – вводятся только три буквы.

INPUT KYPER1 WITH "Введите строку" – выводится подсказка "Введите строку" и запрос на ввод в переменную MYPER1 какого-нибудь значения.

OUTPUT – вывод сообщения на терминал и имеет формат:

OUTPUT < выражение > [тип] [USING < шаблон >]

Здесь: < выражение > – константа, выражение, переменная;

[тип] – тип выводимого выражения (необязателен). Может быть "mm", "str", "log";

using < шаблон > – шаблон выводимого выражения (см команда INPUT).

Примеры:

OUTPUT MYPER1 выводит переменную MYPER1.

OUTPUT MYPER1 USING "aaa" выводит три первых буквенных символа переменной MYPER1.

PERFORM – запуск программы "GURU". Команда имеет формат:

PERFORM "имя файла". Здесь: "имя файла" – файл с командами "GURU", имеющий расширение *.IPF. Пример:

PERFORM MYPRO-GRAM. Запускает программу MYPROGRAM.

RETURN – завершение выполнения процедуры и обеспечение возврата в вызывающую программу или в командный режим. Имеет формат: **RETURN.**

WAIT – останов выполнения программы до тех пор, пока пользователь не нажмет какую-либо клавишу. Имеет формат: **WAIT.**

WHILE-DO – создание цикла, состоящего из других команд "GURU". Цикл обрабатывается до тех пор, пока не будут удовлетворены заданные условия. Имеет формат:

WHILE < условия > DO < утверждения > ENDWHILE.

Здесь: < условия > – критерий, специфицированный в терминах логических выражений;

< утверждение > – последовательность команд "GURU".

TEST – проверка значения специфицированного выражения для определенного варианта. Имеет формат:

TEST < выражение >

CASE < выражение – 1: > < утверждение – 1 >

CASE < выражение – 2: > < утверждение – 2 >

:

OTHERWISE: < утверждение >

ENDTEST.

Здесь: <выражение> – одна или несколько констант и/или переменных, объединенных операторами, которые вычисляют символьное, числовое или логическое значение (TRUE или FALSE);

< утверждение > – последовательность команд "GURU"; OTHERWISE необязательно.

IF-THEN-ELSE – реализует один набор команд, если удовлетворяются указанные условия или другой набор команд, если условия не удовлетворяются. Имеет формат:

IF < условия > THEN < утверждение > ELSE < утверждения > ENDIF.

Здесь: < условия > – критерии, указанные на основе логического выражения;
< утверждения > – набор команд "GURU".
ВНИМАНИЕ! IF-THEN-ELSE не имеет ничего общего с правилами (RULE) "GURU".

CONTINUE – останавливает обработку части DO команды WHILE-DO и повторно оценивает условия WHILE.

BREAK – останов обработки DO-WHILE или обеспечивает выход из TEST.

CLEAR – очистка экрана.

BYE – выход из командного режима.

HOW переменная – вывод информации о том, как система определила значение переменной.

WHY правило – вывод информации об обработке правила.

HELP – вызов подсказки.

SHOW ENVIRONMENT – вывод значений всех переменных среды.

Команды текстового редактора

TEXT <имя файла> – запуск текстового редактора "GURU";

READ <имя файла> – прекращение обработки текущего текста и начало обработки другого текстового файла без остановки и повторного вызова текстового редактора;

TOP – передвинуть курсор в верхнюю часть текущего текста;

BOTTOM – передвинуть курсор в нижнюю часть текущего текста;

GOTO <позиция> – передвинуть курсор в определенную строку текста;

SEARCH – поиск определенной группы символов в тексте;

INSERT <имя файла> – вставка содержимого файла в текущий текст;

PRINT – печать текста;

QUIT – выход из текстового редактора без сохранения изменений;

STOP – выход из текстового редактора с сохранением;

<CTRL-I> – помощь.

ПРИЛОЖЕНИЕ 6. ПОСТАНОВКА ЗАДАЧ НА ЛАБОРАТОРНЫЕ РАБОТЫ

В данном приложении приведены постановки задач для выполнения лабораторных работ по изучению инструментальных возможностей оболочки GURU Индивидуальные варианты заданий для каждой работы, примеры и контрольные вопросы выдаются студентам на каждую лабораторную работу.

1. ТЕМА «ОСНОВНЫЕ ВОЗМОЖНОСТИ ОБОЛОЧКИ GURU»

ЦЕЛЬ:

Формирование знаний и практических умений по использованию возможностей оболочки GURU.

ИСХОДНЫЕ ДАННЫЕ:

Описание и текст программы, эталоны и тесты для заданного варианта программы ЭС.

ЗАДАЧИ:

1. Ознакомиться с описанием работы заданной программы учебной ЭС.
2. Изучить алгоритм функционирования учебной ЭС по ее описанию и тексту программы.
3. Изучить эталоны и тесты для проверки учебной ЭС (п.2.2).
4. Изучить режим диалогового взаимодействия пользователя с оболочкой GURU (п.1.6).
5. Изучить работу подсистемы объяснений оболочки GURU (п.1.4).
6. Запустить заданную программу и проверить правильность работы программы ЭС на эталонных тестах с помощью команд HOW и WHY (п.1.4).
7. Сделать выводы о правильности решения задач учебной ЭС.
8. Оформить и защитить результаты работы у преподавателя.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить функционирование оболочки GURU и ее подсистемы объяснений.
2. Ответить на контрольные вопросы.
3. Решить задачи, перечисленные выше, и оформить результаты работы.
4. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты:

- Текст учебной программы ЭС на входном языке оболочки GURU;
- Выводы по работе.

2. ТЕМА «РАЗРАБОТКА ПРОГРАММЫ УЧЕБНОЙ ЭС»

ЦЕЛЬ:

Формирование знаний и практических умений для разработки программы учебной ЭС.

ИСХОДНЫЕ ДАННЫЕ

Вариант тестового описания ЭС для разработки программы-прототипа ЭС.

ЗАДАЧИ:

1. Изучить команды языка GURU (п.1.6.2 – п.1.6.6).
2. Разработать для заданного варианта ЭС текст программы на языке оболочки GURU.
3. Разработать эталоны и тесты для проверки правильности работы программы ЭС.
4. Создать текст программы ЭС.
5. Отладить и проверить разработанный вариант ЭС на тестах.
6. Сделать выводы о правильности решения задач учебной ЭС.
7. Оформить и защитить результаты работы у преподавателя

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить языковые возможности команд оболочки GURU для разработки программы.
2. Ответить на контрольные вопросы.
3. Решить задачи, перечисленные выше, и оформить результаты работы.
4. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты

- Описание ЭС на естественном языке.
- Описание эталонов и тестов для проверки программы ЭС.
- Исходный текст отлаженной программы ЭС на входном языке оболочки GURU.
- Выводы о результатах разработки и проверки программы ЭС.

3. ТЕМА «ФАКТОРЫ УВЕРЕННОСТИ»

ЦЕЛЬ:

Формирование знаний и практических умений по применению факторов уверенности в программе ЭС.

ИСХОДНЫЕ ДАННЫЕ:

1. Текст и описание программы на языке оболочки GURU.
2. Задание на внесение изменений в текст программы ЭС.
3. Текст созданной программы ЭС (П.6.2.)

ЗАДАЧИ:

1. Изучить представление нечетких и недостоверных знаний (п.1.2.11).
2. Изучить описание заданного варианта программы ЭС.
3. Внести изменения в заданный текст программы с использованием факторов уверенности и отладить измененный вариант программы ЭС.
4. Внести изменения в разработанную программу ЭС (см.П.6.2.) и отладить эту программу.
5. Проверить с помощью команд HOW и WHY правильность работы созданной ЭС.
6. Сделать выводы о правильности решения задач ЭС.
7. Оформить и защитить результаты работы у преподавателя.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить возможности представления нечетких и недостоверных знаний.
2. Ответить на контрольные вопросы.
3. Решить задачи, перечисленные выше, и оформить результаты работы.
4. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты:

- Текст программы ЭС с изменениями;
- Набор эталонов и тестов для проверки правильности работы программы ЭС;
- Выводы о результатах работы.

4.ТЕМА «КОМАНДНЫЙ РЕЖИМ»

ЦЕЛЬ:

Формирование знаний и практических умений по использованию возможностей командного языка оболочки GURU.

ИСХОДНЫЕ ДАННЫЕ:

1. Пример учебной программы и ее описание;
2. Индивидуальный вариант задания для разработки программы.

ЗАДАЧИ:

1. Изучить возможности командного языка оболочки GURU (п. 1.5, приложение 5).
2. Изучить возможности текстового редактора оболочки GURU (приложение 5).
3. Ознакомиться с учебным примером программы.
4. Разработать и отладить программу в соответствии с заданным вариантом.
5. Проверить правильность работы разработанной программы.
6. Сделать выводы о командном режиме оболочки GURU.
7. Оформить и защитить результаты работы у преподавателя.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить возможности командного режима оболочки GURU.
2. Ответить на контрольные вопросы.
3. Разработать, отладить и документировать программу.
4. Продемонстрировать работу программы преподавателю.
5. Решить задачи, перечисленные выше, и оформить результаты работы.
6. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты:

- Программа на языке оболочки GURU;
- Выводы по работе.

5.ТЕМА «ВОЗМОЖНОСТИ ДЛЯ РАБОТЫ С ЭЛЕКТРОННЫМИ ТАБЛИЦАМИ»

ЦЕЛЬ:

Формирование знаний и практических умений по использованию электронных таблиц (ЭТ) в рамках оболочки GURU.

ИСХОДНЫЕ ДАННЫЕ:

1. Пример программы с использованием электронных таблиц.
2. Индивидуальный вариант задания для разработки ЭТ и программы.

ЗАДАЧИ:

1. Изучить режим обработки ЭТ и команды для работы с ЭТ (см. п. 1.2.9, приложение 3).
2. Ознакомиться с учебным примером программы по созданию и работе с ЭТ.
3. Разработать структуру ЭТ для заданного варианта задания.
4. Разработать программу, с помощью которой можно ввести данные в ЭТ.
5. Ввести данные в ЭТ.
6. Рассчитать и распечатать ЭТ.
7. Документировать результаты работы программы.
8. Оформить и защитить у преподавателя отчет по работе.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить режим обработки ЭТ.
2. Ответить на контрольные вопросы.
3. Решить задачи, перечисленные выше, и оформить результаты работы.
4. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты:

- Структура ЭТ;
- Последовательность действий для создания ЭТ;
- Текст программы;
- Распечатка полученных результатов;
- Выводы о результатах работы.

6.ТЕМА «ГРАФИЧЕСКИЕ ВОЗМОЖНОСТИ»

ЦЕЛЬ:

Формирование знаний и практических умений по использованию графических возможностей в рамках оболочки GURU.

ИСХОДНЫЕ ДАННЫЕ:

Индивидуальный вариант задания для разработки графических диаграмм.

ЗАДАЧИ:

1. Изучить команды для работы с графическими средствами.
2. Разработать структуру ЭТ для заданной графической диаграммы и ее вывода.
3. Разработать программу для ввода данных в ЭТ и вывода ЭТ в виде графической диаграммы.
4. Запустить и проверить работу программы.
5. Оформить и защитить у преподавателя отчет по работе.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить теоретическую часть (см. приложение 4).
2. Ответить на контрольные вопросы.
3. Решить задачи, перечисленные выше, и оформить результаты работы.
4. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты:

- Структура ЭТ;
- Определение графической диаграммы;
- Текст программы;
- Выводы о результатах работы.

7.ТЕМА «ПРОЕКТИРОВАНИЕ ЭС»

Лабораторные работы по тематике проектирование прототипа ЭС приведены в [12] и включают следующие работы:

- «Постановка и извлечение знаний»;
- «Структурирование знаний»;
- «Формализация знаний».

Перечисленные работы могут проводиться как в указанной в данном приложении последовательности работ (после изучения инструментальных возможностей оболочки GURU), так и до изучения возможностей оболочки GURU. Последовательность работ определяется организацией учебного процесса по данной дисциплине.

8.ТЕМА «РЕАЛИЗАЦИЯ И ДОКУМЕНТИРОВАНИЕ ЭС»

ЦЕЛЬ:

Формирование знаний и практических умений по реализации и документированию прототипа ЭС.

ИСХОДНЫЕ ДАННЫЕ:

1. Результаты проектирования ЭС по индивидуальному варианту задания.
2. Примеры программных документов для прототипа ЭС.

ЗАДАЧИ:

1. Реализовать спроектированную ЭС.
2. Проверить работу программы ЭС на тестовых наборах и эталонах.
3. Изучить назначение и структуру документов «Текст программы» и «Описание применения».
4. Разработать документы «Текст программы» и «Описание применения» для ЭС.
5. Оформить и защитить у преподавателя отчет по работе.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить назначение и структуру документов «Текст программы» и «Описание применения».
2. Ответить на контрольные вопросы.
3. Решить задачи, перечисленные выше, и оформить результаты работы.
4. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты:

- Листинг исходного текста программы;
- Результаты проверки работы ЭС;
- Документ «Текст программы»;
- Документ «Описание применения».

9.ТЕМА «ИСПЫТАНИЕ ЭС»

ЦЕЛЬ:

Формирование знаний и практических умений по испытанию программы прототипа ЭС.

ИСХОДНЫЕ ДАННЫЕ:

Результаты выполнения лабораторных работ по реализации и документированию прототипа ЭС.

ЗАДАЧИ:

1. Изучить назначение и структуру программного документа «Программа и методика испытаний».
2. Разработать программный документ «Программа и методика испытаний».
3. Продемонстрировать преподавателю процедуру испытаний прототипа ЭС.
4. Оформить и защитить у преподавателя отчет по работе.

ПОРЯДОК ВЫПОЛНЕНИЯ РАБОТЫ:

1. Изучить теоретическую часть.
2. Ответить на контрольные вопросы.
3. Решить задачи, перечисленные выше, и оформить результаты работы.
4. Защитить у преподавателя результаты выполнения лабораторной работы.

РЕЗУЛЬТАТ ВЫПОЛНЕНИЯ РАБОТЫ:

Отчет по данной лабораторной работе включает следующие результаты:

- Документ «Программа и методика испытаний»;
- Результаты испытаний прототипа ЭС.

УЧЕБНОЕ ИЗДАНИЕ

Составители:
Хвещук Владимир Иванович
Муравьев Геннадий Леонидович

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

ПО ДИСЦИПЛИНЕ

«ТЕХНОЛОГИЯ И ИНСТРУМЕНТАЛЬНЫЕ СРЕДСТВА ПРОЕКТИРОВАНИЯ ИНТЕЛЛЕКТУАЛЬНЫХ СИСТЕМ»

Часть 2. Реализация и сопровождение экспертных систем

для студентов специальности «Искусственный интеллект»

Ответственный за выпуск: Хвещук В.И.
Редактор: Строкач Т.В.
Компьютерная верстка: Боровикова Е.А.
Корректор: Никитчик Е.В.

Подписано к печати 24.11.2009 г. Формат 60x84 1/16. Гарнитура «Arial Narrow».
Бумага «Снегурочка». Усл. печ. л. 3,5. Уч.-изд. л. 3,75. Заказ № 1085. Тираж 40 экз.
Отпечатано на ризографе учреждения образования
«Брестский государственный технический университет».
224017, г. Брест, ул. Московская, 267.