Козин Р.Г.

ЭКСПЕРТНЫЕ СИСТЕМЫ

Учебное пособие

Москва
МИФИ, 2008

Экспертные системы. – М.: МИФИ, 2008. – 87 с.

Учебное пособие в конспективной форме знакомит с практическим опытом построения нескольких прототипов экспертных систем, базирующихся на различных формах представления знаний.

Пособие написано по материалам лекций, прочитанных в МИФИ, и предназначено для студентов и специалистов, желающих познакомиться с некоторыми особенностями разработки экспертных систем.

Содержание.

1. Введение в экспертные системы……..4
2. Оболочка экспертной системы с вероятностной базой знаний……9
3. Экспертная система с моделью знаний в виде графа вывода…….18
4. Экспертная система с продукционной моделью знаний…….....33
5. Экспертная система с семантической моделью знаний……..47
6. Модель экспертной системы мониторинга с использованием правил и фреймов…...61
7. Экспертная система планирования последовательности операций робота с использованием предикатов.....64
8. Экспертная система тестирования.....78
Список литературы

Раздел 1. Введение в экспертные системы

Экспертная система (ЭС) – программный комплекс, который, опираясь на знания экспертов в конкретной предметной области (базу знаний – БЗ), дает рекомендации менее квалифицированным специалистам аналогичного профиля, а также любым пользователям.

Предпосылки создания ЭС, **выбор предметной области**:

1. Опыт специалиста существенно помогает при решении задачи, причем наблюдается существенное различие в качестве и времени решения задачи у новичка и специалиста.
2. Есть эксперты, готовые поделиться своим опытом.
3. Часто возникает потребность решения задач подобного рода.
4. Предметная область достаточно хорошо «очерчена».
5. При решении задач необходимо знание эвристик (система логических приемов и правил, о которых говорят: «знаю, что следует делать, но не могу точно сказать почему», или сопровождаемых словами – «как правило, обычно»), используемых экспертами.
6. Решение задачи не требует большого числа арифметических вычислений, а в основном опирается на логические операции с символьными переменными.
7. Используемые при решении задачи входные данные не полные и имеют нечеткие значения, т.е. «многогранны и зашумлены ».
8. Число объектов, явлений, ситуаций и связей между ними слишком велико.
9. Имеются сомнения в достоверности входной и базовой информации.
10. Решение задачи может быть получено лишь с определенной долей уверенности.
11. Есть будущие потенциальные пользователи ЭС.
12. Имеются вычислительная техника и специалисты для разработки ЭС.

Основное отличие ЭС от остальных программных комплексов:

1. ЭС работают со знаниями, которые хранятся в БЗ в виде записей на формальном **языке представления знаний** (ЯПЗ).
2. В обычных программах знания «зашиты» в виде детерминированных алгоритмов

Люди, причастные к ЭС:

1. **Эксперт** – человек, обладающий опытом и знаниями в конкретной предметной области. Желательные качества: доброжелательность, готовность поделиться опытом, умение формулировать свои «знания», заинтересованность в работе (моральная, материальная).
2. **Инженер знаний –** человек, занимающийся «извлечением» знаний, их формализацией и определяющий стратегию работы ЭС. Желательные качества: коммуникабельность, умение ставить наводящие вопросы, способность структурно и логически мыслить.
3. **Программист –** осуществляет программную реализацию ЭС.
4. **Пользователь** – человек, использующий ЭС для получения рекомендаций в конкретной предметной области, определяемой содержанием БЗ.

Классификация ЭС по функциональному назначению:

1. **ЭС для интерпретации данных** – многовариантный анализ данных.
2. **ЭС диагностические** – обнаружение неисправностей, отклонений от нормы.

3. **ЭС мониторинговые** – непрерывная интерпретация данных в реальном масштабе времени с целью обнаружения не штатных ситуаций.
4. **ЭС прогнозирования** – логический вывод вероятных последствий данной аварийной ситуации.
5. **ЭС планирования** – построения плана действий для достижения поставленной цели, например, для интеллектуальных роботов.
6. **ЭС обучения** – управление процессом обучения.
7. **ЭС проектирования –** построение сложного объекта, удовлетворяющего заданным требованиям.

Классификация ЭС по степени интеграции с другими пакетами программ:

1. **Автономные** ЭС – работают самостоятельно, осуществляя консультирование пользователя.
2. **Гибридные** ЭС – встроены в прикладные программные комплексы, которые решают более общие задачи.

Типовая структура автономной ЭС представлена на рис.1.

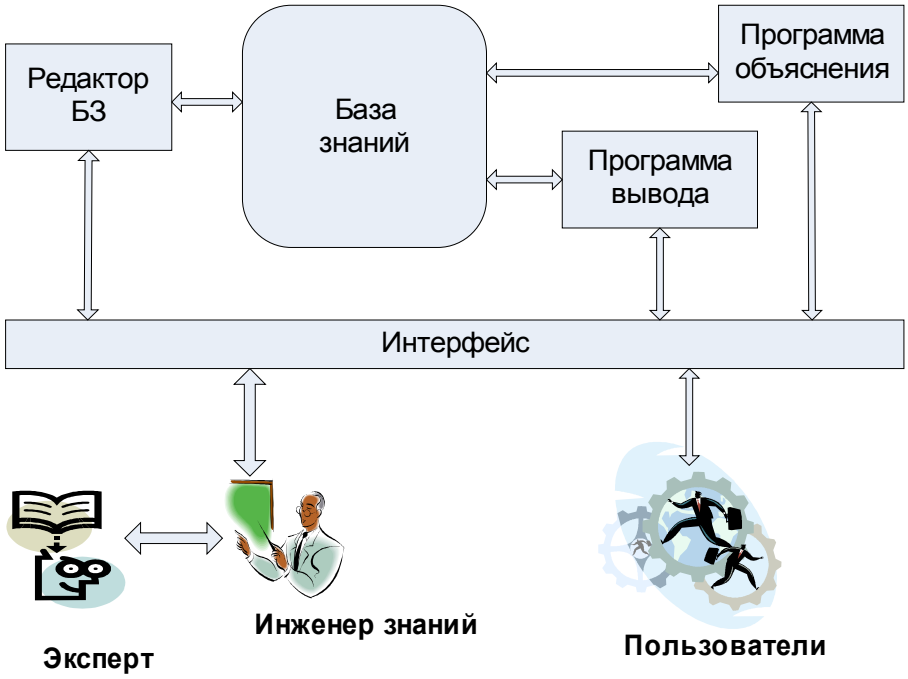


Рис.1 Типовая структура ЭС

Здесь:

Редактор БЗ – позволяет инженеру знаний вводить и редактировать БЗ на входном ЯПЗ, осуществляет проверку правильности кодирования входной информации, транслирует БЗ в формат внутреннего ЯПЗ.

База знаний – совокупность знаний конкретной предметной области (в виде логических правил, графа вывода и т.д.) на внутреннем ЯПЗ.

Программа вывода – блок логического вывода, который реализует принятую стратегию вывода, моделируя рассуждения эксперта.

Программа объяснения – выдает цепочку правил или ответов пользователя, которая привела к выбору предложенной рекомендации.

Интерфейс – обеспечивает «доброжелательный» диалог с программами ЭС.

Этапы разработки ЭС:

1. **Демонстрационный** прототип – используется для демонстрации правильности выбранного подхода в реализации ЭС.
2. **Исследовательский** прототип – предназначен для исследования слабых сторон выбранного подхода и нахождения путей для их устранения.
3. **Действующий** прототип – надежно работает, но «тихоходен» и требует много памяти.
4. **Промышленная** ЭС – решает экспертные задачи быстро и качественно.
5. **Коммерческая** ЭС – промышленная ЭС, хорошо документированная и снабженная сервисным обслуживанием.

Извлечение знаний:

Источники:

Книги	Дискуссии
Диалоги	Интервью
Экспертные игры	Наблюдения
Лекции	Эксперты

Порядок структурирование знаний для заданной предметной области**:**

1. Определение списка терминов.
2. Составление списка основных понятий и их атрибутов.
3. Нахождение отношений между понятиями.
4. Определение структуры входной и выходной информации.
5. Выбор стратегии вывода.
6. Формулирование ограничений выбранной стратегии.

Способы формализации знаний:

1. **Продукционная модель знаний** – знания хранятся в виде правил вида:

«Если (УСЛОВИЯ), то (ЗАКЛЮЧЕНИЕ)»

Заключения делятся на **промежуточные** и **конечные**, завершающие вывод. Например:

Если «давление падает» И «пасмурно», то «будет дождь»
Если «будет дождь», то «следует взять зонтик»

Вывод может быть **прямым** (от заданных исходных условий к конечному заключению), **обратным** (от конечного заключения к проверке условий его подтверждающих) и смешанным. Поиск может выполняться как **«в глубину»** (последовательный анализ каждой ветки в дереве вывода на всю ее глубину), так и **«в ширину»** (при продвижении по графу вывода осуществляется одновременный анализ компонент всех ветвей, расположенных на одной глубине графа).

2. Семантические сети – ориентированный граф, вершины которого **понятия**, а дуги – смысловые **отношения** между ними.

Классификация сетей: по типу отношений - однородные (один тип отношений), неоднородные (различные типы отношений); по количеству связываемых понятий - бинарные (отношения связывают два понятия), n-ортные (отношения связывают более двух понятий).

Основные типы отношений:

- элемент класса
- пример элемента класса
- атрибутивные связи (имеет свойство)
- значение свойства
- часть - целое
- функциональные
- количественные
- пространственные
- временные
- логические (И, ИЛИ, НЕ)

В семантических сетях используется принцип наследования, когда свойства понятия-класса автоматически присваиваются понятиям-экземплярам данного класса. Пример сети приведен на рис.2.

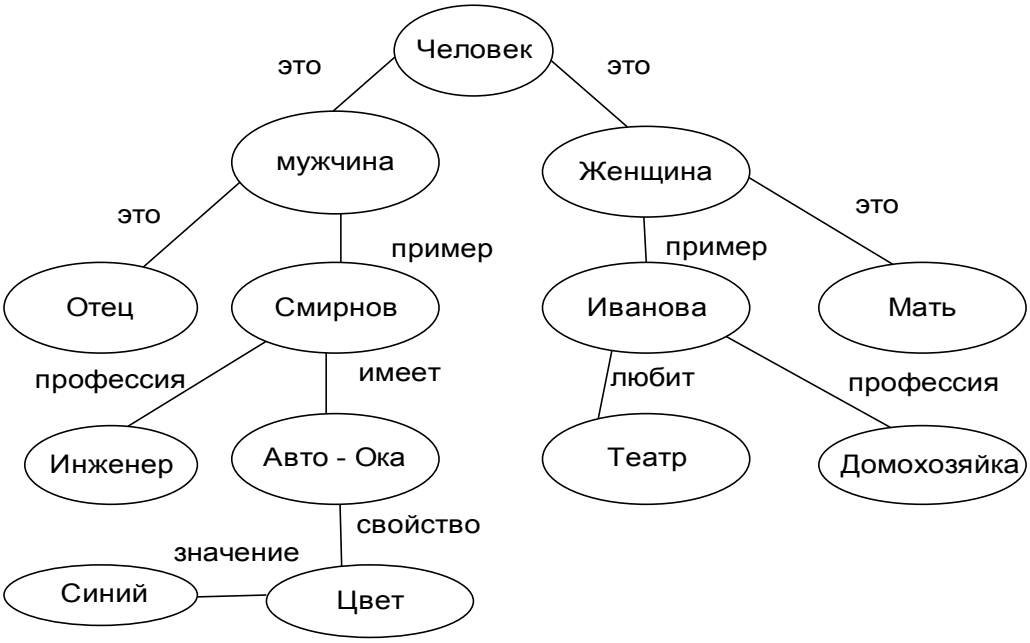


Рис. 2. Пример семантической сети

3. Представление знаний в виде **графа вывода,** в верхних узлах которого расположены конечные заключения, а в ниспадающих – промежуточные заключения, условия-факты, логические и прочие операции. На рис.3 дан фрагмент графа БЗ, диагностирующей характер неисправности принтера.

Рис. 3. Пример графа вывода

4. Вероятностная модель знаний представляет собой исходную совокупность априорных вероятностей заключений-гипотез $P(H_i)$ и условных вероятностей $P(E_j : H_i), P(E_j : не\, H_i)$ свидетельств, связанных с этими гипотезами. Механизм вывода для данной БЗ сводится к последовательному анализу очередного j-го свидетельства (ответ пользователя «да» или «нет») и пересчету апостериорных вероятностей связанных с ним i-ых гипотез по формулам (получены из известной в теории вероятности формулы Байеса)

$$P(H_i : E_j) = \frac{P(E_j : H_i)\,P(H_i)}{P(E_j : H_i)\,P(H_i) + P(E_j : не\, H_i)(1 - P(H_i))}$$

$$P(H_i : не\, E_j) = \frac{(1 - P(E_j : H_i))P(H_i)}{1 - P(E_j : H_i)\,P(H_i) - P(E_j : не\, H_i)(1 - P(H_i))}$$

Принимается, что свидетельства статистически независимы. Поэтому для последующего свидетельства рассчитанные по формулам апостериорные вероятности гипотез рассматриваются как текущие априорные. Вывод завершается после нахождения гипотезы с максимальной вероятностью. Пример такой БЗ: гипотеза 1 – болезнь грипп $P(H_1) = 0{,}4$; свидетельства – высокая температура $P(H_1 : E_1) = 0{,}7$; кашель $P(H_1 : E_2) = 0{,}4$; насморк $P(H_1 : E_3) = 0{,}6$; …

5. Модель знаний в виде **фреймов –** абстрактных образов для представления стереотипов объектов, понятий или ситуаций. Существуют фреймы-шаблоны и фреймы-образцы, описывающие конкретные экземпляры объектов, понятий и ситуаций. Фреймы через свои атрибуты объединяются в сети с наследованием свойств.

Характерная структура фрейма представлена в таблице 1

Таблица 1

Имя фрейма			
Имя слота (атрибута) 1	Значение слота	Способ получения значения	Имя присоединенной процедуры
…	…	…	…

Здесь способы получения значения слота могут быть следующими:
- по умолчанию
- через наследование
- по формуле, указанной в слоте
- через присоединенную процедуру
- из диалога с пользователем
- из базы данных.

Пример фрейма-экземпляра (таблица 2, БД – база данных)

Таблица 2

	Комната		
Тип	жилая	-	-
Номер	108	-	Вызов СУБД
Владелец	-	См. БД	-
Высота потолка	-	См. БД	-
Число окон	-	См. БД	-
Площадь	-	См. БД	-

6. Модель знаний представляющей собой набор **двухместных предикатов**, каждый из которых определяет некоторое отношение между своими операндами. К этим предикатам добавляется набор базовых операторов, которые позволяют производить изменение этих предикатов. С помощью подобной модели обычно реализуются ЭС планирования последовательности операций переводящих исходное состояние мира в некоторое конечное состояние, задаваемое конкретным целевым условием. При этом как состояние мира, так и целевое условие записываются своей группой предикатов.

При создании экспертных систем применяют как специализированные языки (например, Пролог, CLIPS), так и обычные языки программирования. Либо используют пустые экспертные оболочки с встроенными механизмами вывода и объяснения.

Для более детального знакомства с экспертными системами можно воспользоваться литературой, приведенной в конце пособия.

В последующих разделах рассматриваются примеры построения прототипов экспертных систем с различными формами представления знаний.

Тестирование этих прототипов на небольших базах знаний показало их удовлетворительную работоспособность.

Раздел 2. Оболочка экспертной системы с вероятностной базой знаний.

Рассматривается прототип оболочки экспертной системы, работающей с произвольной вероятностной базой знаний, в которой поиск доминирующей гипотезы осуществляется с использованием теоремы Байеса. При создании оболочки использованы идеи, описанные в работе [1]. Оболочка состоит из трех подсистем: подсистема ввода БЗ (транслятор), подсистема вывода доминирующей гипотезы и подсистема объяснения причин выбора предложенной гипотезы.

Вероятностная модель БЗ представляет собой совокупность гипотез (например, болезни) с их исходными априорными вероятностями $p_i \equiv P(H_i)$ и связанных с ними свидетельств (например, симптомы болезней), для которых определены условные вероятности относительно этих гипотез: $p^{+}_{ji} \equiv P(E_j : H_i)$, $p^{-}_{ji} \equiv P(E_j : не\, H_i)$.

Основные требования к БЗ:
- непротиворечивость – любая совокупность свидетельств должна подтверждать не более одной гипотезы;
- презентабельность (полнота) – для всех гипотез максимально возможные вероятности, достигаемые при благоприятных ответах на все связанные с гипотезой свидетельства, должны быть одного порядка.

Ввод БЗ в оболочку ЭС осуществляется на входном языке подсистемы ввода, который имеет следующие операторы:

1. *rem текст* - текст комментария к БЗ на входном языке

2. $\begin{matrix} aim; p; ns \\ !текст \\ [!текст] \\ n_1; p^+; p^- \\ - \ - \ - \ - \ - \\ n_{ns}; p^+; p^- \end{matrix}$ - оператор, задающий гипотезу и связанные с ней свидетельства.

Здесь текст (до двух строчек, «!» – маркер оператора текста) – текст гипотезы; p - априорная вероятность гипотезы; ns - число свидетельств, связанных с данной гипотезой; n_i - порядковый номер свидетельства в БЗ на входном языке; p^+, p^- - условные вероятности данного свидетельства относительно гипотезы (при ее наличии и отсутствии).

3. $\begin{matrix} s \\ !текст \\ [!текст] \end{matrix}$ - оператор, задающий свидетельство с его текстом

Подсистема ввода БЗ:
1. Позволяет сохранять исходную БЗ в виде текстового файла, а также загрузить ее из ранее сохраненного файла.
2. Выполняет диагностику исходной БЗ: проверяет наличия текста, параметров требуемого формата ($0 \le p \le 1, n$ - *целые*) и их количество, правильность ссылок на номера свидетельств.
3. Переводит исходную БЗ во внутреннее преставление, используемое подсистемой вывода. Внутренняя БЗ формируется в четырех массивах: массив текстов, массив гипотез, массив свидетельств, связанных со всеми гипотезами и массив номеров текстов свидетельств. Эту БЗ можно сохранить в двух файлах: текстовом с текстами гипотез и свидетельств и двоичном файле, в который вначале записываются размерности всех массивов, а затем и сами числовые массивы.

Ниже приведен пример БЗ в исходном и внутреннем представлениях.

1. Исходная БЗ

```
rem Почему автомобиль не заводится?
s
!фары не горят
s
!указатель бензина на нуле
s
```

!автомашина стояла под дождем
s
!автомашина давно проходила техобслуживание
s
!стартер не проворачивается
s
!автомашина не заводится
aim;0,1;5
!сел аккумулятор
0;0,99;0
1;0,7;0,05
3;0,5;0,2
4;0,99;0
5;1;0,01
aim;0,05;2
!нет бензина
1;1;0,01
5;0,9;0,02
aim;0,01;3
!отсырело зажигание
2;0,9;0,1
3;0,5;0,25
5;0,9;0,02
aim;0,01;2
!замасленные свечи
3;0,5;0,01
5;0,9;0,02

2. БЗ во внутреннем представлении

!! -- массив текстовых записей -- !!
0 фары не горят
1 указатель бензина на нуле
2 автомашина стояла под дождем
3 автомашина давно проходила техобслуживание
4 стартер не проворачивается
5 автомашина не заводится
6 сел аккумулятор
7 нет бензина
8 отсырело зажигание
9 замасленные свечи
!! -- массив гипотез -- !!
(здесь каждая тройка чисел обозначает: номер текста гипотезы; число связанных с ней свидетельств; ее априорная вероятность)
0 – 6; 5; 0,1
1 – 7; 2; 0,05
2 – 8; 2; 0,01
3 – 9; 2; 0,01
!! -- массив свидетельств, собранных со всех гипотез -- !!

(здесь каждая тройка чисел обозначает: номер свидетельства в массиве свидетельств; его условные вероятности p^+ , p^- , относительно конкретной гипотезы)
0 – 0; 0,99; 0,0
1 – 1; 0,7; 0,05
2 – 3; 0,5; 0,2
3 – 4; 0,99; 0,0
5 – 1; 1,0; 0,01
6 – 5; 0,9; 0,02
7 – 2; 0,9; 0,1
8 – 3; 0,5; 0,25
9 – 5; 0,9; 0,02
10 – 3; 0,5; 0,01
11 – 5; 0,9; 0,02
!! -- массив свидетельств с номерами текстов в массиве текстов -- !!
0 – 0
1 – 1
2 – 2
3 – 3
4 – 4
5 – 5

Экран подсистемы ввода БЗ изображен на рис. 4.

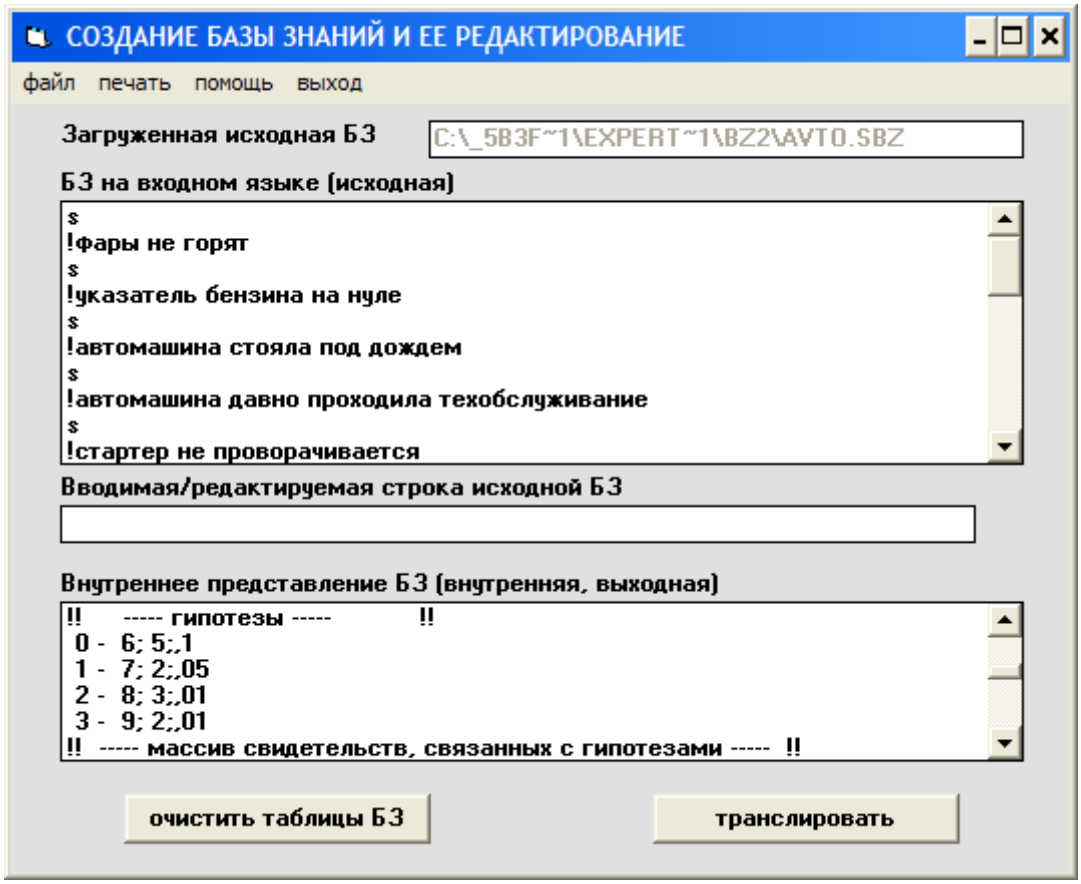


Рис. 4. Форма подсистемы ввода БЗ

Рассмотрим основные элементы стратегии вывода, реализуемой второй подсистемой.

1. В основе стратегии вывода лежит формула Байеса

$$P(H:E) = \frac{P(H \,\&\, E)}{P(E)}$$

которая позволяет уточнить апостериорную условную вероятность гипотезы H при наличии свидетельства E.

С помощью несложных преобразований

$$P(H \,\&\, E) = P(H)P(E:H) = pp^{+}$$
$$P(E) = P(H)P(E:H) + P(\text{не}H)P(E:\text{не}H) = pp^{+} + (1-p)p^{-}$$

эту формулу можно привести к следующему виду

$$P(H:E) = \frac{p^{+}p}{p^{+}p + p^{-}(1-p)} \qquad (1)$$

Здесь справа присутствуют легко определяемые исходные априорные вероятности: $p = P(H),\ p^{+} = P(E:H),\ p^{-} = P(E:\text{не}\,H)$.

Для функционирования системы необходима еще одна формула, которая пересчитывает условную вероятность гипотезы при отсутствии свидетельства (отрицательный ответ пользователя), т.к. в некоторых случаях это может приводить к увеличению условной (текущей апостериорной) вероятности гипотезы

$$P(H:\text{не}\,E) = \frac{(1-p^{+})p}{1-p^{+}p-p^{-}(1-p)} \qquad (2)$$

Предполагается, что свидетельства статистически независимы, поэтому при рассмотрении очередного свидетельства в качестве априорной вероятности гипотезы принимается ее текущая апостериорная вероятность. Поскольку это допущение распространяется на все гипотезы, то оно не должно приводить к большим погрешностям при их ранжировании.

2. Поскольку БЗ имеет вероятностный характер, то пользователю предоставлена возможность определять наличие свидетельства также с некоторой неопределенностью. Он вводит свой ответ q в десятибалльной системе: -5 (твердое НЕТ), 0 (не знаю), 5 (твердое ДА). При этом принимается, что зависимость условной вероятности гипотезы от величины q имеет кусочно-линейную зависимость (см. рис. 5).

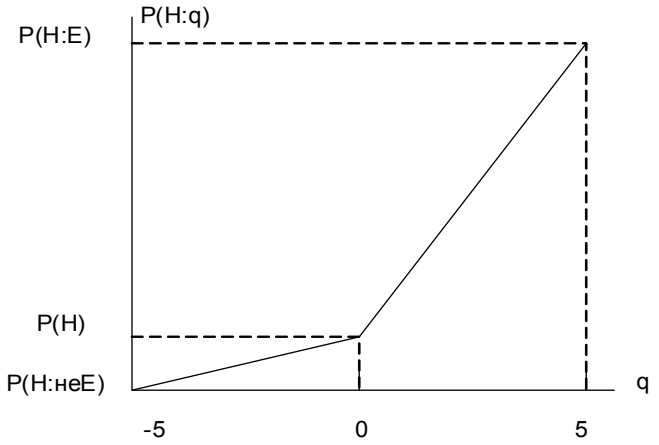


Рис. 5. Зависимость относительной вероятности от значения ответа

Здесь $P(H:E), P(H:не\,E)$ определяются формулами (1), (2), в которых $p = pt \equiv P(H)$ - текущая априорная вероятность гипотезы.

В соответствии с этим графиком после ответа пользователя искомая условная (новая текущая априорная) вероятность гипотезы корректируется по следующим формулам

$$P(H:q) = P(H) + (P(H:E) - P(H))\frac{q}{5}, \quad 0 \le q \le 5$$

$$P(H:q) = P(H:не\,E) + (P(H) - P(H:не\,E))\frac{(5+q)}{5}, \quad -5 \le q < 0$$

Замечание.

Часто пользователю трудно самостоятельно квалифицированно оценить размытую характеристику-свидетельство, например, “высокая температура”. В этом случае, чтобы убрать субъективность в ее оценки, следует использовать **функцию принадлежности,** которая формируется экспертами и устанавливает степень принадлежности конкретного значения, скажем, температуры к размытому понятию “высокая температура”. Например, в данном случае, задав функцию принадлежности в виде таблицы 1, эксперт позволил системе формировать ответ пользователя q автоматически, после ввода пользователем конкретного значения температуры.

Таблица 1

T	36<=	36,5	37	37,5	38	38,5	>=39
q	-5	-3	-1	0	1	3	5

Реализовать такой ввод можно с помощью диалоговой панели, содержащей сам вопрос и соответствующее число управляющих элементов переключателей (см. Рис. 6).

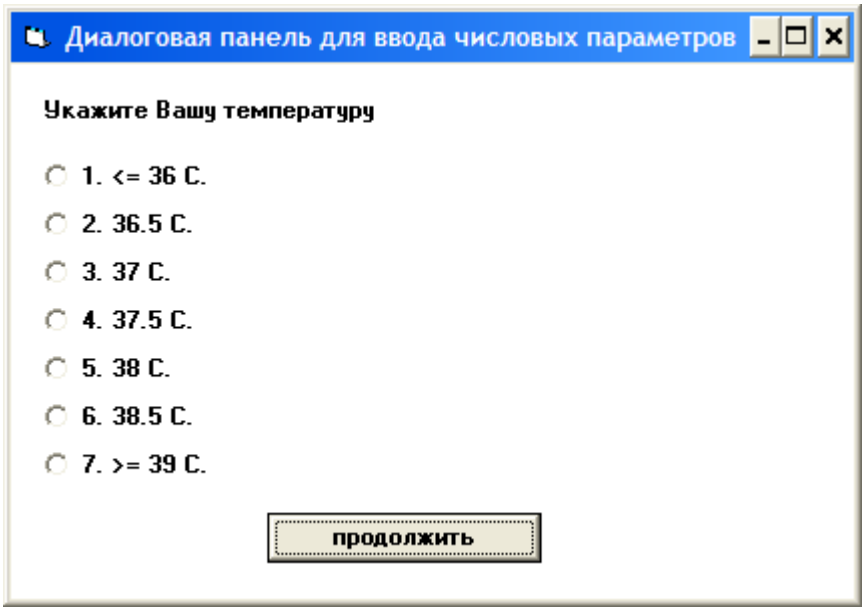


Рис. 6. Возможная форма ввода альтернативного ответа

3. Рассмотрим, каким образом решается проблема выбора очередного свидетельства на обработку. Первоначально для каждого из оставшихся свидетельств рассчитывается его информационная ценность по отношению ко всем оставшимся «рабочим» гипотезам по следующей формуле

$$ЦС = \sum_i \left| P(H_i : E) - P(H_i : не\, E) \right| \qquad (4)$$

где суммирование ведется по всем «рабочим» гипотезам, условная вероятность которых связана с данным свидетельством. Затем на обработку выбирается свидетельство с максимальным значением ЦС. Такой подход порождает стратегию поиска «в ширину».

Приведем алгоритм расчета ценностей свидетельств

1. Цикл по свидетельствам (i)
   2. Если i-ое свидетельство еще не обработано, то
      $ЦС_i = 0$
      3. Цикл по гипотезам (j)
         4. Если гипотеза еще обрабатывается, то
            5. Если в списке свидетельств, связанных с j-ой гипотезой присутствует i-ое свидетельство, то
               $$ЦС_i = ЦС_i + ABS(p(H_j : E_i) - p(H_j : неE_i)$$
            5. Конец если
         4. Конец если
      3. Конец цикла
   2. Конец если
1. Конец цикла

Заметим, что при выборе очередного свидетельства можно использовать и другой критерий. Предъявляется наиболее информативное свидетельство из списка еще не обработанных свидетельств, которые связанны с гипотезой, имеющей наибольшую текущую априорною вероятность. Это соответствует локальной стратегии поиска «в глубину».

4. В процессе вывода дополнительно используются следующие величины: начальные максимально (при благоприятных ответах пользователя) и минимально (при неблагоприятных ответах) достижимые на данной БЗ вероятности гипотез $P_{\max}(H_i), P_{\min}(H_i)$, так и текущие (на свидетельствах, оставшихся к текущему моменту времени) аналогичные вероятности $P_{\max t}(H_i), P_{\min t}(H_i)$.

Поскольку начальные вероятности являются частным случаем текущих, то все указанные величины вычисляются по единому алгоритму (в предположении статистической независимости свидетельств)

1. Цикл по не отброшенным «рабочим» гипотезам
   $p\max t = p\min t = pt$
   2. Цикл по оставшимся свидетельствам

$$py = \frac{p^+ p\max t}{p^+ p\max t + p^- (1 - p\max t)}$$

$$pn = \frac{(1 - p^+) p\max t}{1 - p^+ p\max t - p^- (1 - p\max t)}$$

3. Если $py > pn$, то

$$p\max t = py$$

$$p\min t = \frac{(1 - p^{+})\,p\min t}{1 - p^{+}\,p\min t - p^{-}\,(1 - p\min t)}$$

иначе

$$p\max t = pn$$

$$p\min t = \frac{p^{+}\,p\min t}{p^{+}\,p\min t + p^{-}\,(1 - p\min t)}$$

3. Конец если

2. Конец цикла по свидетельствам

1. Конец цикла по гипотезам

5. При выводе используются следующие эвристики.

1. Если после очередного пересчета всех вероятностей найдена доминирующая гипотеза

$$p\min t_k \geq p\max t_i \quad для\ \forall\, i, где\ i \neq k$$

то k-ая гипотеза предъявляется пользователю и процесс вывода заканчивается (т.к. ее вероятность не зависимо от ответов пользователя уже не может быть меньше вероятности любой другой гипотезы).

2. Если для некоторой гипотезы

$$pt_k < 1.1 p\min_k$$

то k- ая гипотеза «отбраковывается» (становиться не «рабочей» и исключается из дальнейшего рассмотрения).

3. Если для некоторой гипотезы

$$pt_k < 0{,}99 p\max_k$$

то k-ая гипотеза предъявляется пользователю и процесс вывода заканчивается (данная гипотеза «выжила» из БЗ почти все возможное; это правильно, если БЗ - презентабельная).

5. Если для k - ой гипотезы все ее свидетельства обработаны и

$$pt_k \geq p\max t_i \quad для\ \forall\, i, где\ i \neq k$$

то данная гипотеза предъявляется пользователю и процесс вывода заканчивается.

Для обеспечения простоты манипулирования всеми этими величинами в программе вывода используются массивы трех структур:

1. Структура, связанная с конкретной гипотезой (массив $g(ng)$, ng – число гипотез)

nt	Номер текста гипотезы
ns	Число свидетельств, связанных с гипотезой
fl	Флаг =-1 – гипотеза отвергнута, 0 – обрабатывается, 1 – уже обработана
p	Исходная априорная вероятность гипотезы
$p\max$	Исходная максимально достижимая на БЗ вероятность гипотезы
$p\min$	Исходная минимально достижимая на БЗ вероятность гипотезы
pt	Текущая априорная вероятность гипотезы
$p\max t$	Текущая максимально достижимая вероятность гипотезы на оставшихся свидетельствах
$p\min t$	Текущая минимально достижимая вероятность

гипотезы на оставшихся свидетельствах

2. Структура, связанная с конкретным свидетельством (массив $s(ns)$, ns - число свидетельств)

nt	Номер текста свидетельства
fl	Флаг = 0 – свидетельство не обработано, 1 – уже обработано
c	Текущая ценность свидетельства - ЦС
ANS	Ответ пользователя (от -5 до 5)

3. Структура, используемая в массиве ($sg(nsg)$, nsg - размерность этого массива), в котором собраны свидетельства, связанные со всеми гипотезами БЗ

ns	Номер свидетельства в массиве $s(ns)$
py	p^+ - исходная условная вероятность свидетельства при наличии данной гипотезы
pn	p^- - исходная условная вероятность свидетельства при отсутствии данной гипотезы
pyt	$P(H:E)$ - текущая условная вероятность гипотезы при наличии данного свидетельства
pnt	$P(H:\text{не}\,H)$ - текущая условная вероятность гипотезы при отсутствии данного свидетельства

За изменениями компонент рабочих массивов во время вывода можно наблюдать с помощью трех таблиц, представленных на главном экране оболочки (см. рис. 7).

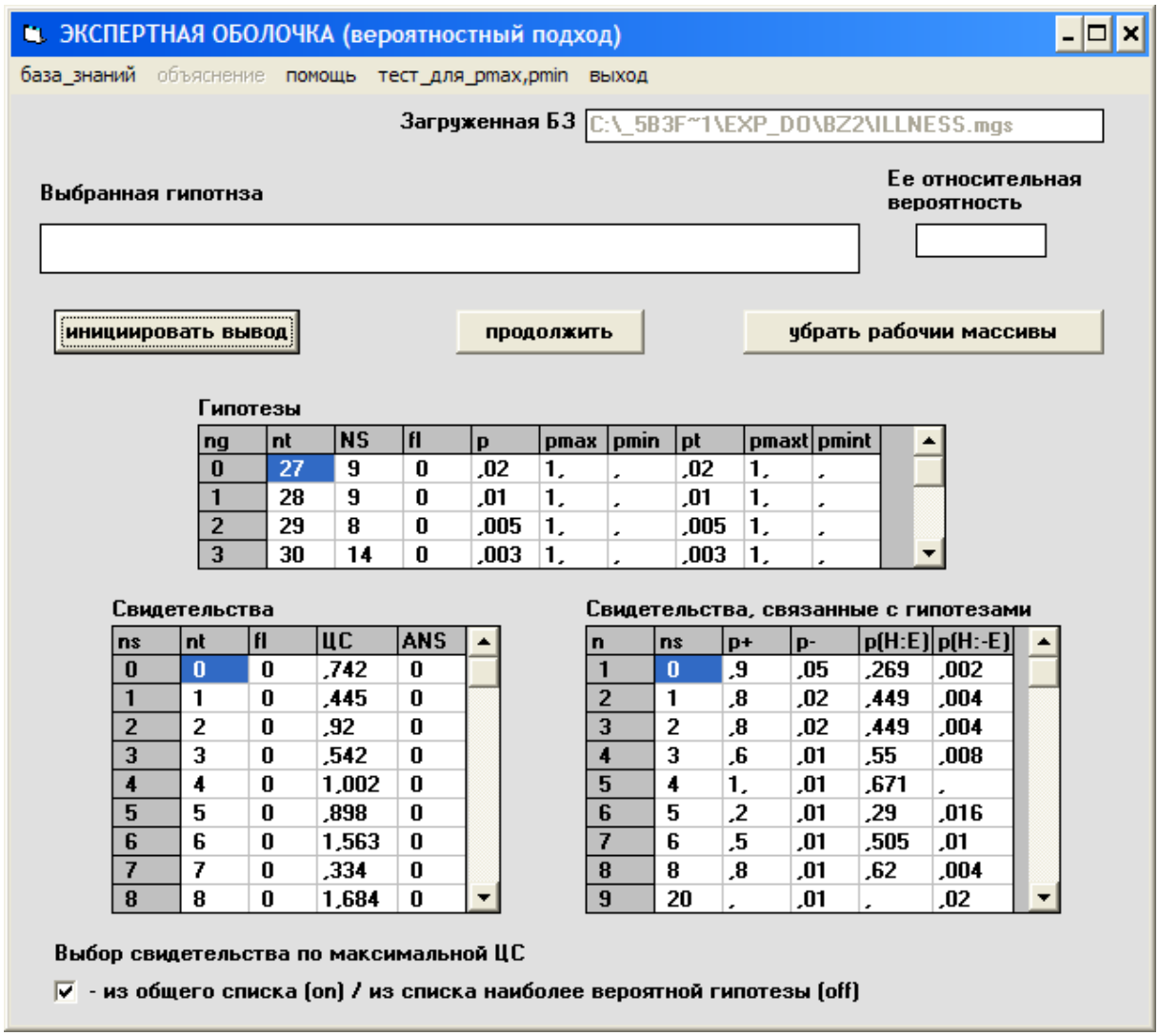


Рис. 7. Главная форма ЭС

Ответ пользователя вводится с помощью вспомогательной панели (рис. 8).

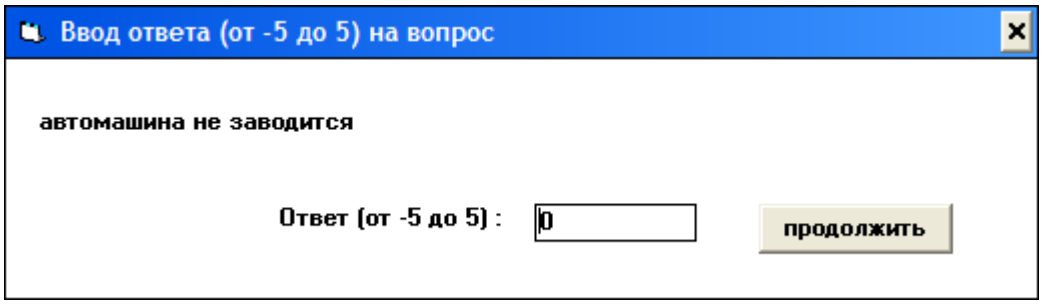


Рис. 8. Диалоговая панель для ввода ответа

Приведем алгоритм работы программы вывода.

1. **Инициализация вывода** (управляющая кнопка «инициировать вывод»):
   - присваиваем нулевые значения флажкам в массиве гипотез и свидетельств;
   - для всех гипотез рассчитываем $p\max$, $p\min$;
   - для всех свидетельств определяем ЦС.

2. **Вывод** (управляющая кнопка «продолжить»):

1. Находим свидетельство с максимальным значением ЦС.
2. Выдаем пользователю выбранное свидетельство и принимаем его ответ в интервале от -5 до 5.
3. Пересчитываем у всех необработанных гипотез (у которых $fl = 0$) текущие значения вероятностей: $p\max t,\ p\min t,\ pt$.

При этом:

- Если у гипотезы $pt \le 1{,}1\,p\min$, то она отбраковывается и для нее устанавливается: $fl = -1,\ p\max t = p\min t = pt$.
- Если у гипотезы $pt \le 0{,}99\,p\max$, то для нее устанавливается: $fl = 1,\ p\max t = p\min t = pt$, она выдается пользователю и процесс вывода завершается.
- Если у k - ой гипотезы не осталось необработанных свидетельств, то для нее устанавливается: $fl_k = 1,\ p\max t_k = p\min t_k = pt_k$. И если при этом выполняется условие $pt_k \ge p\max t_i \quad для\ \forall\, i, i \ne k$, то она ($k$-ая гипотеза) предъявляется пользователю и процесс вывода заканчивается

4. Проверяем наличие доминирующей гипотезы:

Если найдена гипотеза, для которой $p\min t_k \ge p\max t_i \quad для\ \forall\, i, i \ne k$,

то она (k-ая гипотеза) предъявляется пользователю и процесс вывода заканчивается

5. Если все свидетельства уже предъявлены, то

Если у всех гипотез $fl = -1$, то

выдаем сообщение «Нет достоверной гипотезы»

иначе

выдаем гипотезу с максимальным значением относительной вероятности $\dfrac{pt}{p\max}$

конец если

Процесс вывода заканчиваем

конец если

6. Для всех необработанных свидетельств (у которых $fl = 0$) находим ЦС и переходим на пункт 1.

Подсистема объяснения собирает и выдает пользователю (рис. 9) тексты предъявленных свидетельств, которые связаны с выбранной гипотезой, и его ответы на них.

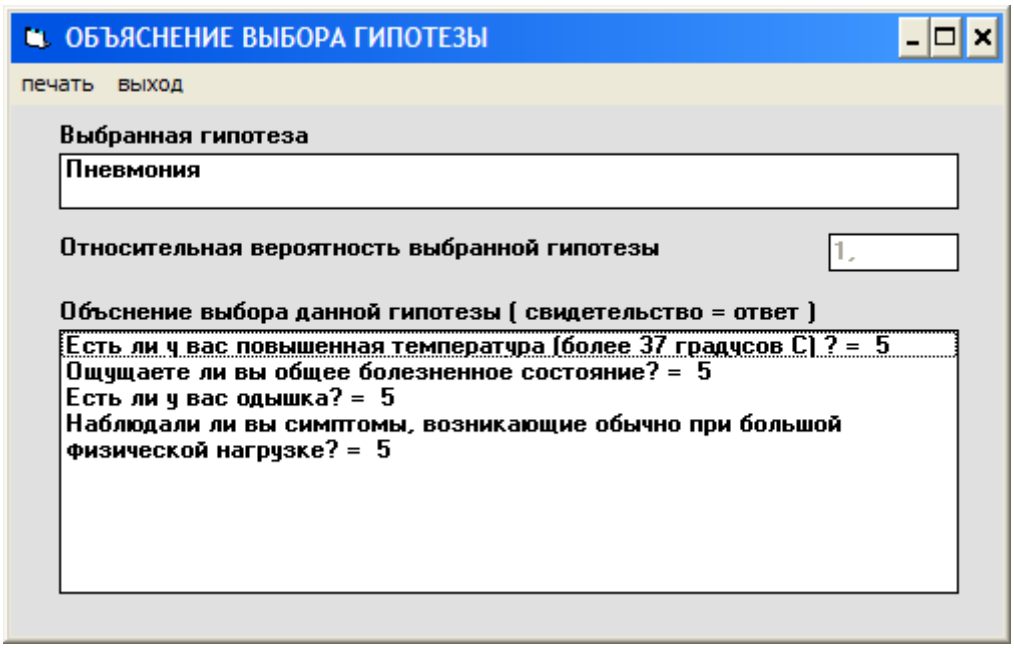


Рис. 9. Форма подсистемы объяснения

Раздел 3. Экспертная система с моделью знаний в виде графа вывода

Обсуждается прототип экспертной оболочки, работающей с произвольной базой знаний в виде графа вывода, построенного с использованием конкретной совокупности типов узлов. В отличие от [2] количество типов узлов, используемых при построении графа, существенно увеличено. Это позволило представлять в виде графа не только продукционные знания, но и знания в виде тестов с альтернативными ответами, как с баллами, так и без них.

Оболочка состоит из трех подсистем: подсистема ввода графа (транслятор), подсистема вывода заключения и подсистема объяснения причин выбора предъявленного заключения.

С первой подсистемой работает инженер знаний. Он вводит граф, кодируя его на входном языке, а подсистема транслирует его во внутреннее представление и сохраняет в виде трех файлов - текстового и двух двоичных файлов с массивом целей-заключений и массивом узлов графа. Одновременно эта подсистема проводит следующую диагностику введенной информации:

- наличие или отсутствие текста у узла
- использование вместо цифры другого символа
- присутствие ссылки на отсутствующий узел
- дублирование номера узла в графе
- не существующий идентификатор узла
- дублирование номеров нисходящих узлов у данного узла
- в узле сравнения значение нижней границы интервала меньше значения верхней

Приведем алгоритм программы, которая тестирует и переводит исходную БЗ во внутреннее представление.

nc=0, nt=0, nmaxg=-1 (инициализация счетчиков числа гипотез, текстовых записей и максимального номера узла графа вывода)
1. Цикл по записям исходной БЗ
Выделяем первый символ записи – s
Если s=”;”, то это текст и поэтому nt=nt+1
Если s=”r”, то это комментарий – rem и поэтому пропускаем запись
Если s=”a”, то это цель и поэтому
nc=nc+1
Если за ней не стоит текстовая запись, то формируем сообщение об ошибке
Конец если
Если первый символ это число - n, то
Если n>nmaxg, то nmaxg=n
Выделяем первый символ идентификатора узла – j
Если j =(q, r или v), то
Если далее нет текстовой записи, то формируем сообщение об ошибке
Конец если
Иначе (это не число)
Формируем сообщение об ошибке
Конец если
1. Конец цикла
Если были ошибки, то выдаем сформированные сообщения и заканчиваем трансляцию
2. Цикл по записям исходной БЗ
Если это “aim”, то
Последовательно выделяем Ц и n1.
Если это не числа, то формируем сообщение об ошибке
Конец если
Если это узел графа, то
Выделяем первый символ идентификатора узла – j
Если это неизвестный идентификатор, то
Формируем сообщение об ошибке
Иначе
Для данного конкретного идентификатора выделяем его параметры
Если это не числа, то формируем сообщение об ошибке
Конец если
2. Конец цикла
Если были ошибки, то выдаем сформированные сообщения и заканчиваем трансляцию
3. Цикл по сформированному массиву узлов графа
Проверка, что для каждого узла графа номера нисходящих узлов указывают на существующие узлы графа
Проверка отсутствия дублирования номеров нисходящих узлов у данного узла
Проверка, что в узле сравнения значение нижней границы меньше верхней
При нарушении условия формируется соответствующее сообщение об ошибке
3. Конец цикла
Если были ошибки, то выдаем сформированные сообщения и заканчиваем трансляцию

Со второй и третьей подсистемами работает пользователь, желающий получить рекомендации, генерируемые конкретной базой знаний.

Подсистема вывода реализует обратную цепочку рассуждений (от целей к фактам). Цели-заключения анализируются (со связанными с ними частями графа) по очереди до

тех пор, пока не будет найдено заключение с максимальной истинностью (с максимальным логическим значением).

Подсистема объяснения собирает и выдает пользователю тексты тех узлов графа, которые согласно ответам пользователя обеспечили максимальное значение истинности выбранному заключению. При этом используется прямая стратегия объяснении (от фактов к заключению).

Поскольку экспертная система работают с нечеткими знаниями, то узлы графа наделяются ценность (значимостью) в установлении истинности заключения (от 60% до 100%), а ответы пользователя на вопросы из базы могут быть любым действительным числом от 0 (твердое НЕТ) до 1 (твердое ДА).

В графе вывода допускается использование следующих типов узлов. Указаны правила их кодирования на входном языке транслятора и порядок обработки при выводе и объяснении. Этот порядок необходимо учитывать при построении графа вывода):

- ЦЕЛЬ-ЗАКЛЮЧЕНИЕ - aim,n1
;текст
(;текст)

где n1 – номер первого узла графа, связанного с данной цель-заключением; текст – текст для данного узла (допускается до 2-х строк длиной не более 69 символов каждая, все строки должны начинаться символом «;» и не могут иметь на конце знака «+»).
при выводе и объяснении:
- загружает n1 в рабочий стек, который используется для обработки графа. Узел на обработку выбирается из вершины стека, а узел, требующий обработки, предварительно помещается в вершину стека. Таким образом, осуществляется перемещение по узлам графа.

- КОММЕНТАРИЙ - rem текст

- СООБЩЕНИЕ - N,r
;текст
(;текст)

где N – номер самого узла.
при выводе:
- выдает свой информационный текст пользователю;
- присваивает своему логическому значению 0 (всегда включается в граф через узел «или» первым номером);
- убирает свой номер из рабочего стека.
при объяснении:
- удаляет свой номер из рабочего стека.

- ВОПРОС - N,q,Ц
;текст
(;текст)

где Ц – ценность-значимость данного узла в % (целое число от 0 до 100).
при выводе:
- выдает текст вопроса пользователю;
- принимает ответ в виде вещественного числа от 0 (твердое НЕТ) до 1 (твердое ДА) и вычисляет свое логическое значение по формуле

$$l = 0{,}01 * Ц * ответ$$

- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;
- удаляет свой номер из рабочего стека.

- ЛОГИЧЕСКОЕ «И» - N,and,Ц,n1,n2(,n3)
(;текст)
(;текст)

где ni – номера нисходящих узлов.
при выводе:
- по очереди загружает в рабочий стек номера нисходящих узлов (если после обработки узел возвращается с логическим значением 0, то обработка оставшихся узлов прекращается) и после их обработки присваивает своему баллу максимальный балл этих узлов, а свое логическое значение вычисляет по формуле

$$l = 0{,}01 * Ц * \min(l(ni))$$

где l(ni) – логическое значение ni-го узла
- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;
- убирает свой номер из рабочего стека;
- если над ним в графе стоит узел «нет», то в рабочий стек загружает номер нисходящего узла с минимальным логическим значением, иначе – номера всех нисходящих узлов.

- ЛОГИЧЕСКОЕ «ИЛИ» - N,or,Ц,n1,n2(,n3)
(;текст)
(;текст)

при выводе:
- по очереди загружает в рабочий стек номера нисходящих узлов (если после обработки узел возвращается с логическим значением 1, то обработка оставшихся узлов прекращается) и после их обработки свое логическое значение вычисляет по формуле

$$l = 0{,}01 * Ц * \max(l(ni))$$

- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;
- в рабочий стек вместо своего номера загружает номер узла с максимальным логическим значением.

- ЛОГИЧЕСКОЕ «НЕТ» - N,not,Ц,n1
(;текст)
(;текст)

при выводе:
- загружает в рабочий стек номер нисходящего узла и после его обработки свое логическое значение вычисляет по формуле

$$l = 0{,}01 * Ц * (1-l(n1))$$

- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;

- на свое место в рабочий стек загружает «-1» и сбрасывает fl=0 – индикатор прохождения через узел «нет». При обратном прохождении в рабочем стеке через «-1» индикатор восстанавливается fl=1, а сама единица удаляется из рабочего стека.

- КОРРЕКЦИЯ - N,%,Ц,n1
(;текст)
(;текст)

при выводе:
- загружает в рабочий стек номер нисходящего узла и после его обработки свое логическое значение вычисляет по формуле

$$l = 0{,}01 * Ц * l(n1)$$

- убирает свой номер из рабочего стека.

при объяснении:
- загружает свой номер в стек выдачи текстов;
- в рабочий стек вместо своего номера загружает номер нисходящего узла.

- АЛЬТЕРНАТИВА - N,v(,bal)
;текст
(;текст)

где bal – балл присвоенный альтернативе.

при выводе:
- самостоятельно не анализируется (только совместно с узлами «ВЫБОР АЛЬТЕРНАТИВЫ»).

при объяснении:
- загружает свой номер в стек выдачи текстов;
- удаляет свой номер из рабочего стека.

- ВЫБОР АЛТЕРНАТИВЫ - N,set,n1,n2(n3,n4,n5)

при выводе:
- выдается пронумерованный список текстов альтернатив + «Введите номер выбранной альтернативы»;
- после ответа пользователя, выбранной альтернативе присваивается логическое значение 1, а оставшимся альтернативам и самому узлу 0 (всегда включается в граф через узел «или» первым номером);
- убирает свой номер из рабочего стека.

при объяснении:
- убирает свой номер из рабочего стека.

- ВЫБОР АЛЬТРНАТИВЫ с учетом ее балла -
N,bset,n1,n2(,n3,n4)
(;текст)
(;текст)

при выводе:
- выдает свой текст (если он есть) + пронумерованный список текстов альтернатив + «Введите номер выбранной альтернативы»;
- после ответа пользователя, выбранной альтернативе и самому узлу присваивается логическое значение 1, а остальным альтернативам 0;
- баллу самого узла присваивает балл выбранной альтернативы или ее порядковый номер в списке (если она не имеет балла);
- убирает свой номер из рабочего стека.

при объяснении:
- загружает номер выбранной альтернативы и свой номер в стек выдачи текстов;
- убирает свой номер из рабочего стека.

- СУММИРОВАНИЕ БАЛЛОВ - N,msum,n1,n2(,n3,n4,n5)
при выводе:
- по очереди загружает в рабочий стек номера нисходящих узлов и после их обработки присваивает своему баллу сумму баллов этих узлов, а своему логическому значению значение 1;
- убирает свой номер из рабочего стека.
при объяснении:
- убирает свой номер из рабочего стека;
- загружает номера нисходящих узлов в рабочий стек;

- ОПЕРАТОРЫ СРАВНЕНИЯ -

N,eq,numb,n1	? – numb = bal(n1)
lt	? – numb > bal(n1)
gt	? – numb < bal(n1)
N,cmp,numb1,n1,numb2	? – numb1 < bal(n1) < numb2
(;текст)	
(;текст)	

где numb,numb1,numb2 – целые числа, bal(n1) – балл узла n1.
при выводе:
- если нисходящий узел не обработан, то добавляет его номер в рабочий стек, иначе проверяет выполнение соответствующего условия и при его истинности присваивает своему логическому значению 1, иначе 0;
- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;
- в рабочий стек на свое место загружает номер нисходящего узла.

- КОРРЕКЦИЯ БАЛЛА у узла графа -

N,put,bal,n1
(;текст)
(;текст)

при выводе:
- загружает в рабочий стек номер нисходящего узла и после его обработки устанавливает рабочим массивам:

l(N) = l(n1)
Если l(n1)=1, то bal(N)=bal, иначе bal(N)=0

- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;
- в рабочий стек вместо своего номера загружает номер нисходящего узла.

- ДОПОЛНИТЕЛЬНОЕ ЛОГИЧЕСКОЕ «И» -

N,iand,Ц,n1,n2(,n3)
(;текст)
(;текст)

где ni – номера нисходящих узлов.

при выводе:
- по очереди загружает в рабочий стек номера нисходящих узлов (если после обработки узел возвращается с логическим значением 0, то обработка оставшихся узлов прекращается) и после их обработки присваивает своему баллу максимальный балл этих узлов, а свое логическое значение вычисляет по формуле

$$l = 0{,}01 * Ц * \min(l(ni))$$

где l(ni) – логическое значение ni-го узла
- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;
- убирает свой номер из рабочего стека;
- в рабочий стек вместо своего номера загружает номер узла с максимальным логическим значением.

- ОПЕРАТОР СРАВНЕНИЯ баллов двух узлов -

N,dgt,n1,n2 ? – bal(n1) < bal(n2)
(;текст)
(;текст)

где bal(ni) – балл узла ni.
при выводе:
- если какой-либо нисходящий узел не обработан, то добавляет его номер в рабочий стек;
- после обработки узлов n1 и n2 проверяет выполнение условий: l(n2)=1 и bal(n1) < bal(n2), и при их истинности присваивает своему логическому значению 1, иначе 0;
- убирает свой номер из рабочего стека.
при объяснении:
- загружает свой номер в стек выдачи текстов;
- в рабочий стек на свое место загружает номер последнего нисходящего узла.

- СПИСОК ВОПРОСОВ, оформленных как альтернативы -

N,jset,n1,n2(n3,n4,n5)

при выводе:
- выдается пронумерованный список текстов вопросов;
- после ответа пользователя, выбранным вопросам присваивается логическое значение 1, а остальным вопросам и самому узлу 0 (всегда включается в граф через узел «или» первым номером);
- баллам выбранных вопросов присваивается балл вопроса-альтернативы или 1, если у альтернативы балл отсутствует, а баллам остальных вопросов – 0;
- убирает свой номер из рабочего стека.
при объяснении:
- убирает свой номер из рабочего стека.

Данный узел позволяет построить граф с прямой цепочкой вывода, когда вначале обрабатываются все факты, а затем последовательно анализируются гипотезы

- ВЫЧИСЛЕНИЕ логического значения узла -

N,fact,bal,n1
(;текст)
(;текст)

при выводе:

- загружает в рабочий стек номер нисходящего узла и после его обработки устанавливает:
  если l(n1)=0, то l(N)=0 иначе l(N)=bal(n1)/bal
- убирает свой номер из рабочего стека.

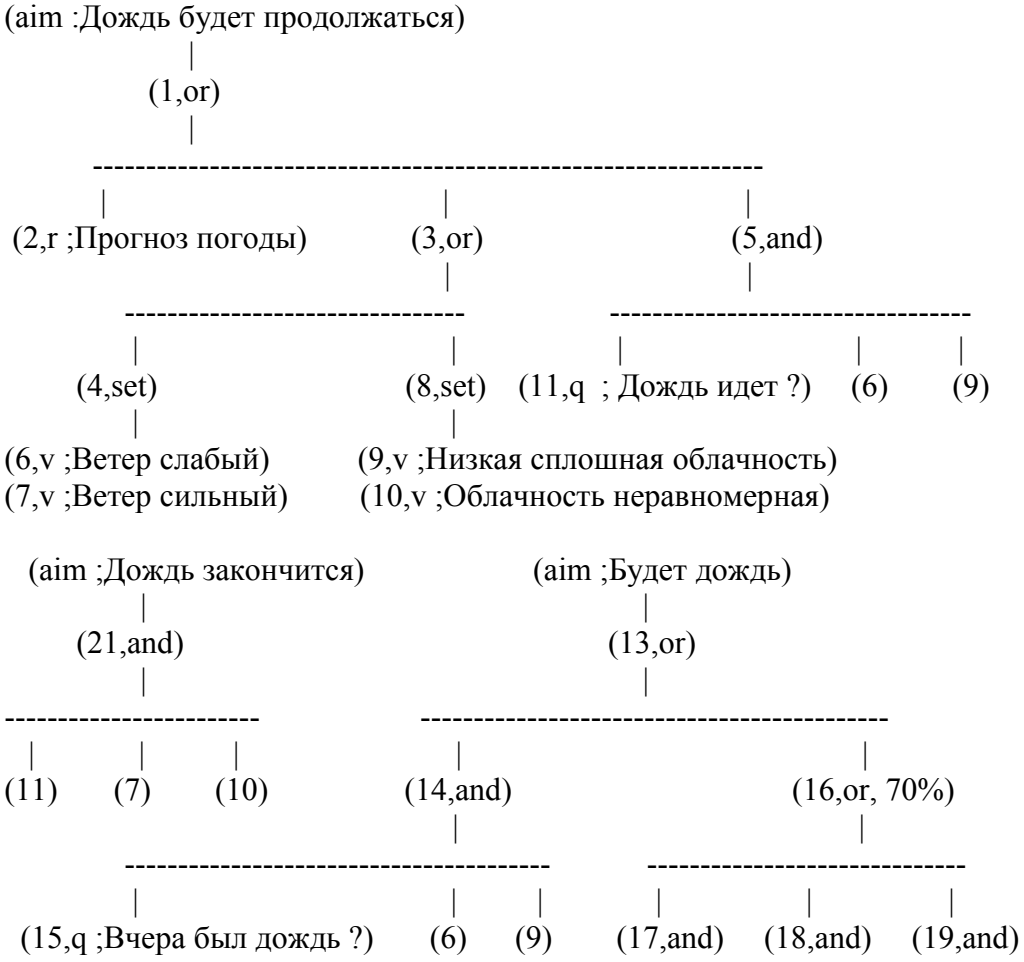
при объяснении:
- загружает свой номер в стек выдачи текстов;
- в рабочий стек вместо своего номера загружает номер нисходящего узла.

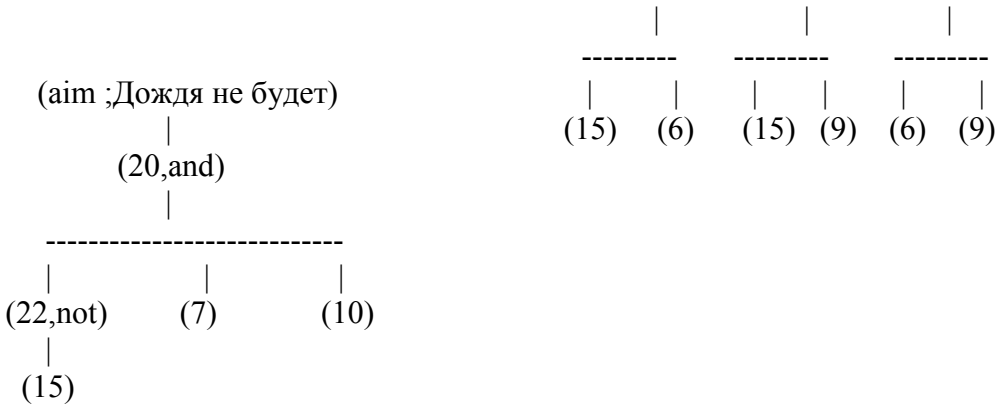
Приведем несколько примеров составления графа вывода.

Пример 1. Продукционная база знаний для предсказания погоды.

ЕСЛИ «идет дождь» И «низкая равномерная облачность» И «ветер слабый», ТО «дождь будет идти» с вероятность 100%
ЕСЛИ «идет дождь» И «облачность неравномерная» И «ветер сильный», ТО «дождь кончится» с вероятность 100%
ЕСЛИ «вчера был дождь» И «низкая плотная облачность» И «ветер слабый», ТО «будет дождь» с вероятностью 100%
ЕСЛИ («вчера был дождь» И «низкая равномерная облачность») ИЛИ («вчера был дождь» И «ветер слабый»,) ИЛИ («низкая равномерная облачность» И «ветер слабый»), ТО «будет дождь» с вероятностью 70%

Построенный граф вывода:





Его кодирование на входном языке системы:

```
aim,1
;Дождь будет продолжаться !
1,or,100,2,3,5
2,r
;Прогноз погоды
3,or,100,4,8
4,set,6,7
5,and,100,11,6,9
6,v
;Ветер слабый
7,v
;Ветер сильный
8,set,9,10
9,v
;Низкая сплошная облачность
10,v
;Облачность неравномерная
11,q,100
;Дождь идет ?
aim,21
;Дождь закончиться !
aim,13
;Будет дождь !
13,or,100,14,16
14,and,100,15,6,9
15,q,100
;Вчера был дождь ?
16,or,70,17,18,19
17,and,100,15,6
18,and,100,15,9
19,and,100,6,9
aim,20
;Дождя не будет !
20,and,100,22,7,10
21,and,100,11,7,10
22,not,100,15
```

На выходе транслятора этот граф представляется в виде трех массивов (каждый массив сохраняется в своем файле: имя.tex, имя.aim, имя.grf, где имя – уникальное имя базы знаний):

- текстовые записи -
0 ;Дождь будет продолжаться !
1 ;Прогноз погоды
2 ;Ветер слабый
3 ;Ветер сильный
4 ;Низкая сплошная облачность
5 ;Облачность неравномерная
6 ;Дождь идет ?
7 ;Дождь закончиться !
8 ;Будет дождь !
9 ;Вчера был дождь ?
10 ;Дождя не будет !

- массив гипотез –
0 – 1, 0
1 – 21, 7
2 – 13, 8
3 – 20,10

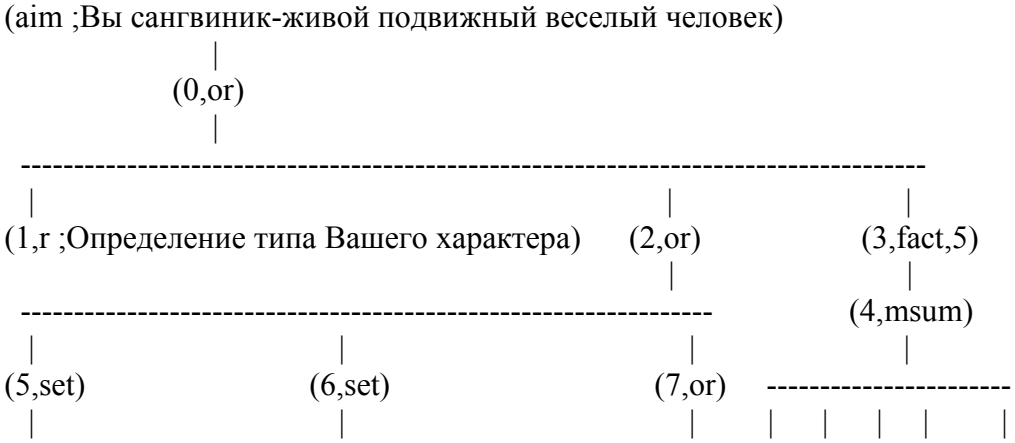
- массив с узлами графа вывода –
(первое число – соответствующий числовой идентификатор узла)
0 - -1, -1, -1, -1, -1, -1 - всюду -1, т.к. узла с таким номером нет
1 - 1, 100, -1, 2, 3, 5
2 - 5, -1, 1, -1, -1, -1
3 - 1, 100, -1, 4, 8, -1
4 – 8, 6, 7, -1, -1, -1
5 – 2, 100, -1, 11, 6, 9
6 – 9, -1, 2, -1, -1, -1
7 – 9, -1, 3, -1, -1, -1
8 – 8, 9, 10, -1, -1, -1
9 – 9, -1, 4, -1, -1, -1
10 – 9, -1, 5, -1, -1, -1
11 – 4, 100, 6, -1, -1, -1
12 - -1, -1, -1, -1, -1, -1 - всюду -1, т.к. узла с таким номером нет
13 – 1, 100, -1, 14, 16, -1
14 – 2, 100, -1, 15, 6, 9
15 – 4, 100, 9, -1 -1, -1
16 – 1, 70, -1, 17, 18, 19
17 – 2,100, -1, 15, 6, -1
18 – 2, 100, -1, 15, 9, -1
19 – 2, 100, -1, 6, 9, -1
20 – 2, 100, -1, 22, 7, 10
21 – 2, 100, -1, 11, 7, 10
22 – 3, 100, -1, 15, -1, -1

Пример 2. Базу знаний в виде теста.

Тест для определения типа Вашего характера.
В каждом варианте выберите одну из альтернатив:
а)
1. на грубость отвечаете шуткой
2. на грубость отвечаете агрессивно
3. на грубость отвечаете спокойно
4. на грубость отвечаете пассивно
б)
1. преобладающее настроение веселое
2. преобладающее настроение не ровное
3. преобладающее настроение ровное
4. преобладающее настроение унылое
в)
1. в компании стараетесь быть в центре внимания
2. в компании ведете себя не ровно
3. в компании ведете себя спокойно
4. в компании смущаетесь, чувствуете себя скованно
г)
1. за любое дело принимаетесь легко и выполняете его с удовольствием
2. легко зажигаетесь на новое дело, но скоро к нему остываете
3. любое дело выполняете добросовестно и до конца
4. за любое дело беретесь с сомнением
д)
1. легки на подъем
2. можете неожиданно (по настроению) сорваться из дома
3. домосед
4. предпочитаете покидать дом только по необходимости

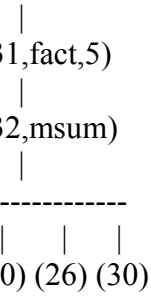
Если больше первых ответов, то Вы сангвиник - живой подвижный веселый человек, вторых - Вы холерик - легко возбудимый агрессивный человек, третьих - Вы флегматик - спокойный малоподвижный солидный человек, четвертых - Вы меланхолик - подавленный мрачный человек.

Граф вывода для данного теста (для большей объективности в каждом варианте – а, б, в, г, д альтернативы поставлены в произвольном порядке)

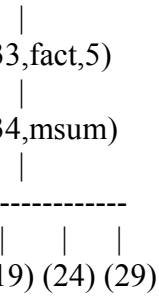


```
(8,v ;текст а-1)      (12,v ;текст б-2)     ----------   (8) (15) (22) (25) (27)
(9,v ;текст а-2)      (13,v ;текст б-4)      |          |       |
(10,v ;текст а-3)     (14,v ;текст б-3)   (16,set)  (17,set)  (18,set)
(11,v ;текст а-4)     (15,v ;текст б-1)      |          |          |
                              (19,v ;текст в-3)  (23,v ;текст г-4)  (27,v ;текст д-1)
                              (20,v ;текст в-2)  (24,v ;текст г-3)  (28,v ;текст д-4)
                              (21,v ;текст в-4)  (25,v ;текст г-1)  (29,v ;текст д-3)
                              (22,v ;текст в-1)  (26,v ;текст г-2)  (30,v ;текст д-2)
```

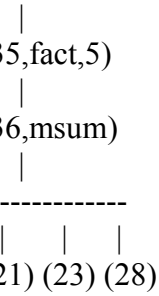
(aim ;Вы холерик-легко возбудимый агрессивный человек)



(aim ;Вы флегматик-спокойный малоподвижный солидный человек)



(aim ;Вы меланхолик-подавленный мрачный человек)



Запись этого графа на входном языке:

```
aim,0
;Вы сангвиник-живой подвижный веселый человек !
0,or,100,1,2,3
1,r
;Определение типа Вашего характера
2,or,100,5,6,7
3,gt,3,4
4,msum,8,15,22,25,27
5,set,8,9,10,11
```

```
6,set,12,13,14,15
7,or,100,16,17,18
8,v
;на грубость отвечаете шуткой
9,v

;на грубость отвечаете агрессивно
10,v
;на грубость отвечаете спокойно
11,v
;на грубость отвечаете пассивно
12,v
;преобладающее настроение не ровное
13,v
;преобладающее настроение унылое
14,v
;преобладающее настроение ровное
15,v
;преобладающее настроение веселое
16,set,19,20,21,22
17,set,23,24,25,26
18,set,27,28,29,30
19,v
;в компании ведете себя спокойно
20,v
;в компании ведете себя не ровно
21,v
;в компании смущаетесь, чувствуете себя скованно
22,v
;в компании стараетесь быть в центре внимания
23,v
;за любое дело беретесь с сомнением
24,v
;любое дело выполняете добросовестно и до конца
25,v
;за любое дело принимаетесь легко и выполняете его с удовольствием
26,v
;легко зажигаетесь на новое дело, но скоро к нему остываете
27,v
;легки на подъем
28,v
;предпочитаете покидать дом только по необходимости
29,v
;домосед
30,v
;можете неожиданно (по настроению) сорваться из дома
aim,31
;Вы холерик- легко возбудимый агрессивный человек !
31,gt,3,32
32,msum,9,12,20,26,30
```

```
aim,33
;Вы флегматик - спокойный малоподвижный солидный человек !
33,gt,3,34
34,msum,10,14,19,24,29
aim,35
;Вы меланхолик -подавленный мрачный человек !
35,gt,3,36
36,msum,11,13,21,23,28
```

Пример 3. Определение психологических типов (факторы оцениваются баллами, указывающих на степень их важности при определении конкретного типа).

1. Напористо-агрессивный

1. Во всем любит конкуренцию – 5
2. Что-либо предлагает энергично и напористо – 5
3. Испытывает чувство собственного преимущества – 5
4. Не терпит неудач и потерь – 10
5. Честолюбив – 3
6. Полон решимости во всем – 3

2. Подчеркнуто любезный

1. Не настойчив в отстаивании своего мнения – 10
2. Считает, что защита лучше нападения – 5
3. В работе стремится избегать осложнений – 5
4. Постоянно демонстрирует сердечность и доброту – 5
5. Пытается со всеми соглашаться – 3
6. Стремится быть полезным – 3

3. Самоуверенный

1. Во всем любит конкуренцию – 3
2. Легко принимает решение – 5
3. Открыт и честен с коллегами – 10
4. Полон решимости во всем – 3
5. Что-либо предлагает энергично и напористо – 5
6. Не всегда прислушивается к чужому мнению - 5

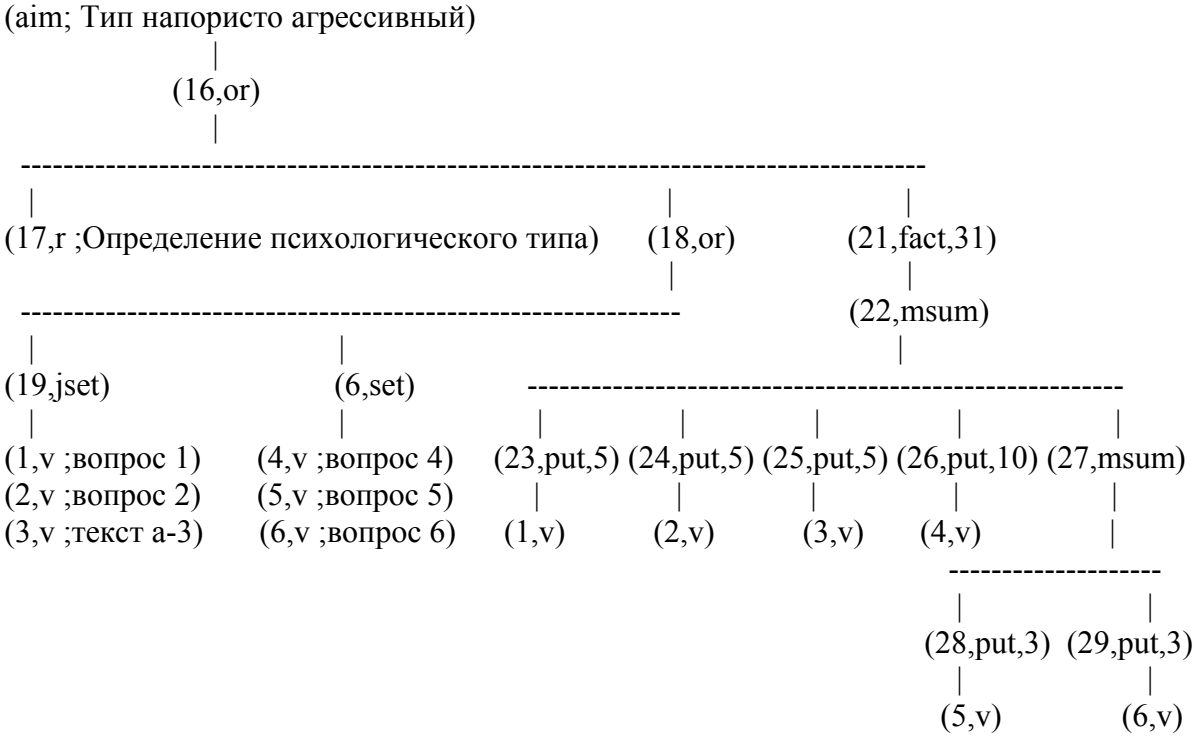
Формулировка вопросов-альтернатив для пользователя:

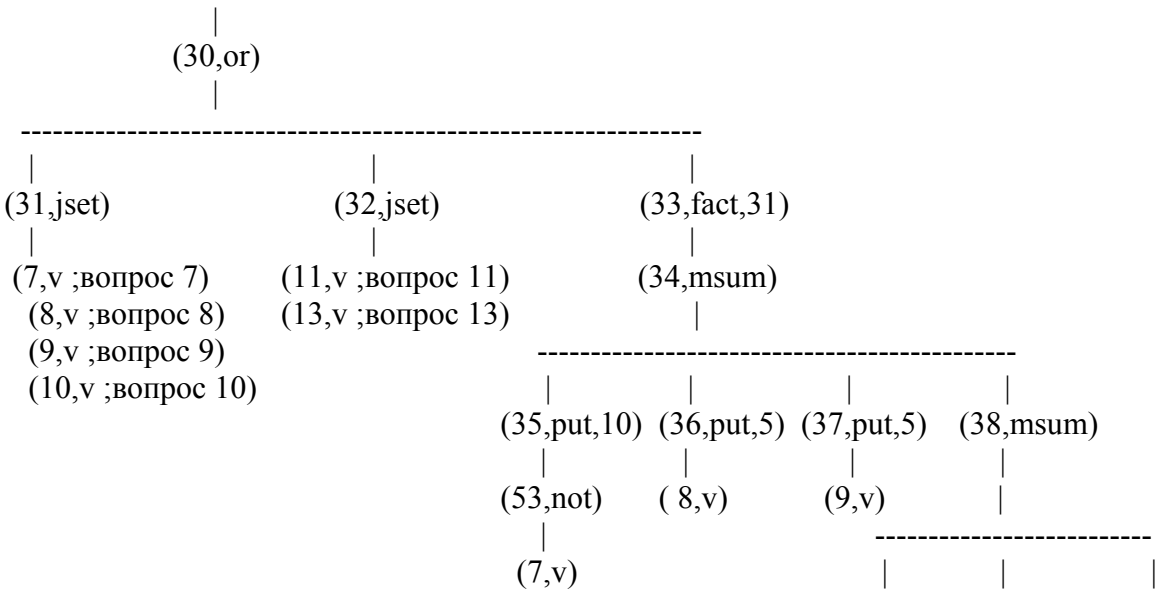
1.Любите ли во всем конкуренцию ?
2. Продвигаете ли свои идеи энергично и напористо ?
3. Испытываете ли чувство собственного превосходства ?
4. Не терпите неудач и потерь ?
5. Вы честолюбивы ?
6. Полны ли решимости во всем ?
7. Настойчивы ли в отстаивании своего мнения ? (Нет идет в плюс для 2-го типа)
8. Считаете ли, что защита лучше нападения ?
9. Стремитесь ли избегать осложнений в общении с коллегами ?

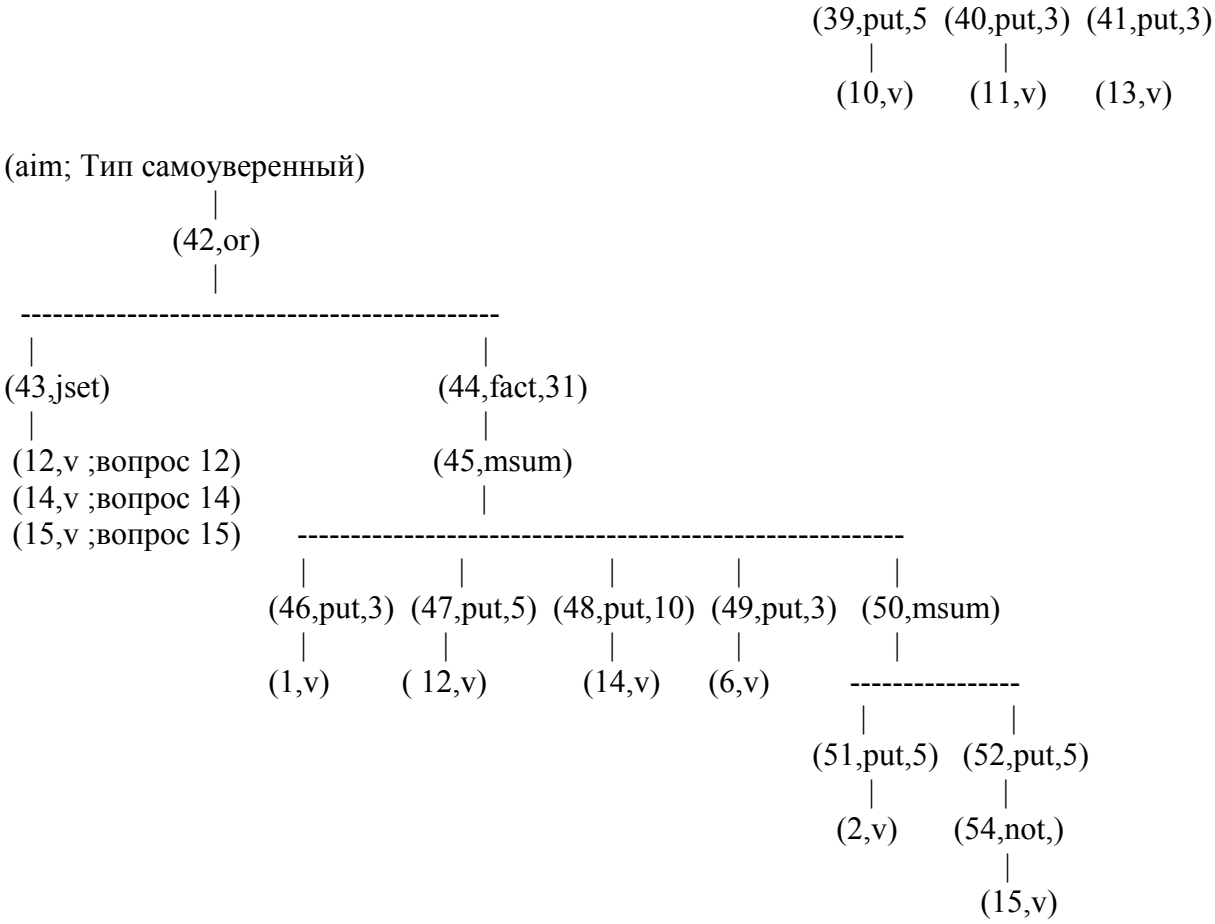
10. Постоянно ли стремитесь демонстрировать сердечность и доброту ?
11. Стремитесь ли со всеми соглашаться ?
12. Легко ли принимаете решения ?
13. Стремитесь ли всегда быть полезным ?
14. Всегда ли открыты и честны с коллегами ?
15. Всегда ли прислушиваетесь к чужому мнению ?

Критерий выбора гипотезы: Чем выше суммарный весовой коэффициент гипотезы, тем ближе характер человека к соответствующему типу.

Граф вывода для данного теста.







Кодировка этого графа вывода на входном языке системы выглядит следующим образом.

```
1,v
;Любите ли во всем конкуренцию ?
2,v
;Продвигаете ли свои идеи энергично и напористо ?
3,v
Испытываете ли  чувство собственного превосходства ?
4,v
;Не терпите неудач и потерь ?
5,v
;Вы честолюбивы ?
6,v
;Полны ли решимости во всем ?
7,v
;Настойчивы ли в отстаивании своего мнения ? (Нет идет в плюс для 2-го типа)
8,v
;Считаете ли,  что защита лучше нападения ?
9,v
;Стремитесь ли избегать осложнений в общении с коллегами ?
10,v
;Постоянно ли стремитесь демонстрировать сердечность и доброту ?
11,v
```

```
;Стремитесь ли со всеми соглашаться ?
12,v
;Легко ли принимаете решения ?
13,v
;Стремитесь ли всегда быть полезным ?
14,v
;Всегда ли открыты и честны с коллегами ?
15,v
;Всегда ли прислушиваетесь к чужому мнению ?
aim,16
;Тип напористо агрессивный
16,or,100,17,18,21
17,r
;Определение психологического типа
18,or,100,19,20
19,jset,1,2,3
20,jset,4,5,6
21,fact,31,22
22,msum,23,24,25,26,27
23,put,5,1
24,put,5,2
25,put,5,3
26,put,10,4
27,msum,28,29
28,put,3,5
29,put,3,6
aim,30
;Тип подчеркнуто  любезный
30,or,100,31,32,33
31,jset,7,8,9,10
32,jset,11,13
33,fact,31,34
34,msum,35,36,37,38
35,put,10,53
53,not,100,7
36,put,5,8
37,put,5,9
38,msum,39,40,41
39,put,5,10
40,put,3,11
41,put,3,13
aim,42
;Тип самоуверенный
42,or,100,43,44
43,jset,12,14,15
44,fact,31,45
45,msum,46,47,48,49,50
46,put,3,1
47,put,5,12
48,put,10,14
```

```
49,put,3,6
50,msum,51,52
51,put,5,2
52,put,5,54
54,not,100,15
```

Для пояснения работы программы вывода приведем алгоритмы нескольких ее компонент.

Используются следующие обозначения:
nc – число целей; ng – максимальный номер узла в графе вывода; p – указатель рабочего стека; ii – числовой идентификатор анализируемой цели; jj – номер анализируемого узла графа; i2 – текущий номер цели с наибольшей вероятностью; i3 – номер текущей анализируемой цели; v – текущее наибольшее значение вероятности (0-1) проанализированных целей (в начале -1); w – пороговое значение вероятности (0-1) для выбора наиболее вероятной гипотезы; c(nc,3) – массив целей (номер нисходящего узла, номер связанного текста, вероятность цели); s(ng) – массив рабочего стека; g(ng,6) – массив узлов графа; l(ng) – массив с логическими значениями узлов графа (-1 по умолчанию, иначе >=0); bal(ng) – массив для хранения баллов, присвоенных узлам графа при его обработке (-1 по умолчанию, иначе >=0).

Алгоритм главной программы вывода:

```
инициализация массивов : c(i,2) = -1, l(i) = -1, bal(i) = -1
p= 0, v= -1
for i3= 0,nc-1   (цикл по гипотезам)
   s(p)= c(i3,0)   (загрузка в стек первого узла подграфа)
   while p≥ 0 do   (обработка связанных узлов подграфа)
      j= s(p)
      if l(j)≥ 0 then p= p-1   (выгружаем узел из стека)
      else
         ii = g(j,0)   (идентификатор анализируемого узла)
         Select case ii   (альтернативный анализ узлов)
            case 1   (анализ узла "И")
            case 2   (анализ узла "ИЛИ")
            - - - - - -
            case 14   (анализ узла "GT")
         end select
      end if
   end while
   p = p + 1, j = s(p)
   if l(j) > v then v = l(j), i2 = i3 end if
   if w ≤ v then
      exit for   (конец вывода)
   end if
end for
if v > 0 then
   Выдаем i2 - ю гипотезу
else
  Выдаем сообщение "Достоверная гипотеза не найдена!"
end for
```

Алгоритм подпрограммы обработки узла «or - ИЛИ» (его компоненты в массиве g(ng,6):

ID	Ц	nt	n1	n2	n3

где ID=1 - числовой идентификатор узла, Ц- ценность узла, nt – номер связанного текста, n1, n2, n3 – номера нисходящих узлов; nt/n3 = -1 – если текст или номер отсутствуют).
Стратегия его обработки: если встретился нисходящий узел с $l = 1$, то остальные узлы уже не анализируются. Здесь номер самого узла j.

```
lm = −1
for i = 3,5
   n = g(j,i)   (номер текущего нисходящего узла)
   if n < 0 then exit for end if
   lt = l(n)   (логическое значение узла)
   if lt < 0 then lm = lt, nm = n, exit for end if
   if lt > lm then lm = lt, nm = n end if
   if lm ≈ 1 then exit for end if
end for
if lm < 0 then   (загружаем в стек номер нисходящего узла)
   p = p + 1, s(p) = nm
else   (убираем ИЛИ из стека, вычисляем его логическое l)
   p = p − 1
   l(j) = lm * g(j,1) * 0,01
end if
```

Алгоритм подпрограммы обработки узла «bset – выбор альтернативы с учетом их баллов» (его компоненты в массиве g(ng,6):

ID	nt	n1	n2	n3	n4

где ID=10 - числовой идентификатор узла, nt – номер связанного текста, n1, n2, n3, n4 – номера нисходящих узлов-альтернатив; nt/n3/n4 = -1 – если текст или номера отсутствуют). Номер самого узла j. Компоненты альтернатив в массиве g(ng,6) (ID=9):

ID	балл	nt	-1	-1	-1

```
nv = 0   (первоначальное число альтернатив у bset)
n= g(j,1)   (номер текста узла bset)
if n≥ 0 then   (текст присутствует)
   выводим текст t(n) на диалоговую панель
end if
for i= 2,5   (цикл по нисходящим альтернотивам)
   n= g(j,i)   (номер очередной альтернативы)
   if n< 0 then exit for end if   (больше нет альтернатив)
   nv= nv+ 1, n= g(n,2)   - номер текста альтернативы
   выводим на диалоговую панель:'nv'+ '.'+ t(n)
end for
выводим на диалоговую панель:
"Введите номер выбранной альтернативы"
возврат из диалоговой панели(проверяет, что 0< n_in≤ nv)
n_in= введенный номер выбранной альтернативы
for i= 2,nv+ 1   (цикл по нисходящим альтернотивам)
   n= g(j,i)
   if i= n_in+ 1 then l(n)= 1 else l(n)= 0 bal(n)= 0 end if
end for
l(j)= 1   (логическое значение узла bset)
n= g(j,n_in+ 1)   (номер узла выбранной альтернативы)
nn= g(n,1)   (исходный балл у выбранной альтернативы)
if nn= - 1 then   (исходный балл отсутствует)
   bal(j)= n_in   (присваеваем балл bset в массиве баллов)
   bal(n)= 1   (присваеваем балл выбранной альтернативы)
else
   bal(j)= nn   (присваеваем балл bset в массиве баллов)
   bal(n)= nn   (присваеваем балл выбранной альтернативы)
end if
p = p - 1   (удалаем узел bset из рабочего стека)
```

Программа объяснения работает с двумя стеками. Первый используется для движения по графу от выбранной гипотезы, второй - для сбора номеров узлов, тексты которых необходимо выдать при объяснении. Этот стек позволяет обеспечить прямой вывод при объяснении – от причин к следствию. При сборе информации порядок обработки узлов графа осуществляется в соответствии с правилами, приведенными ранее при описании порядка кодировки конкретных узлов графа на входном языке транслятора.

Ниже приведены основные экраны данной оболочки.

1. Главный экран (рис. 9). Помимо прочего на нем можно наблюдать за порядком загрузки узлов графа в рабочий стек в процессе вывода. При желании эту функцию можно отключить кнопкой «**убрать рабочий стек**».

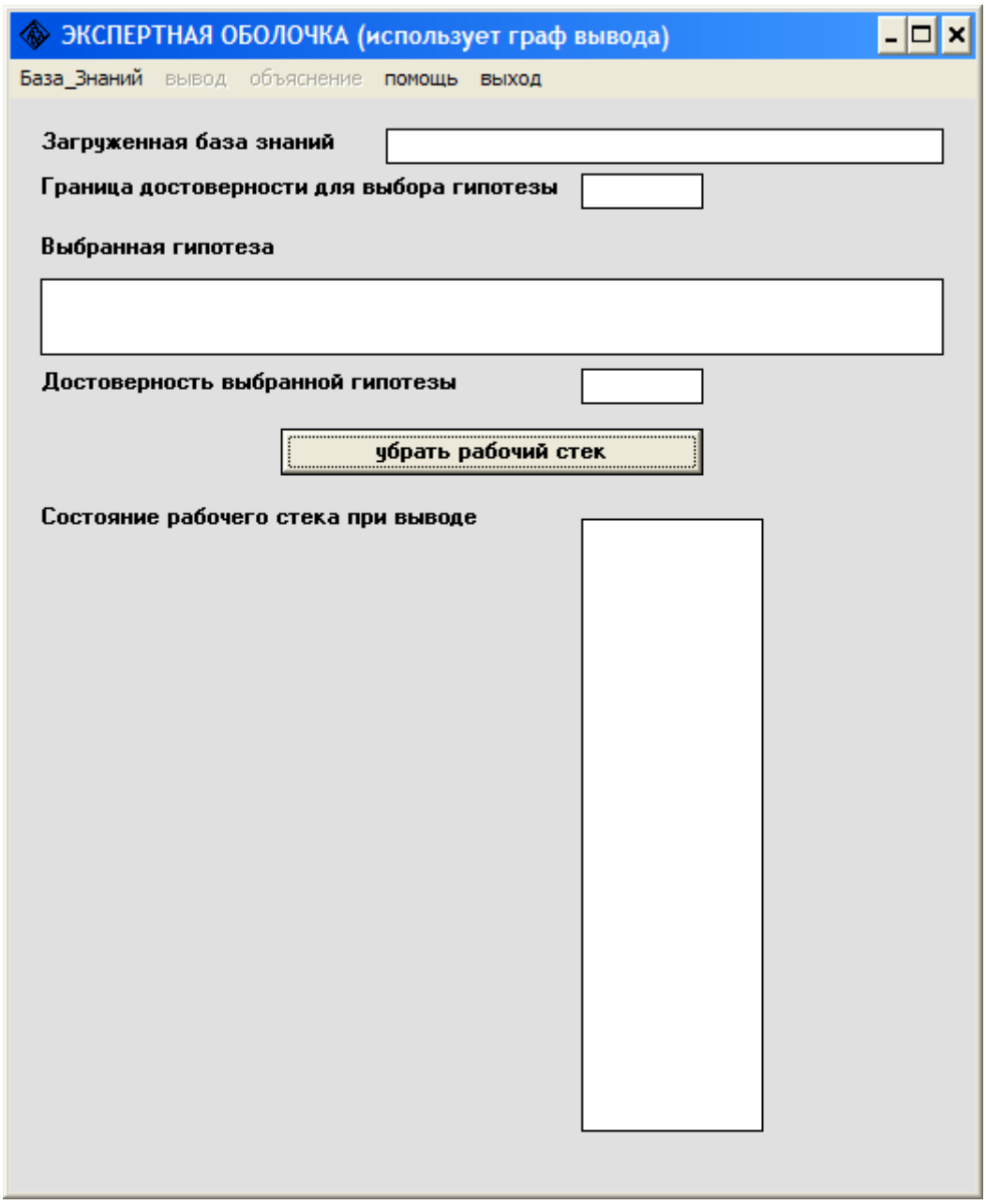


Рис. 9. Главная форма ЭС

2. Экран подсистемы ввода или редактирования базы знаний (дерева вывода) и преобразования ее во внутреннее представление (рис. 10)

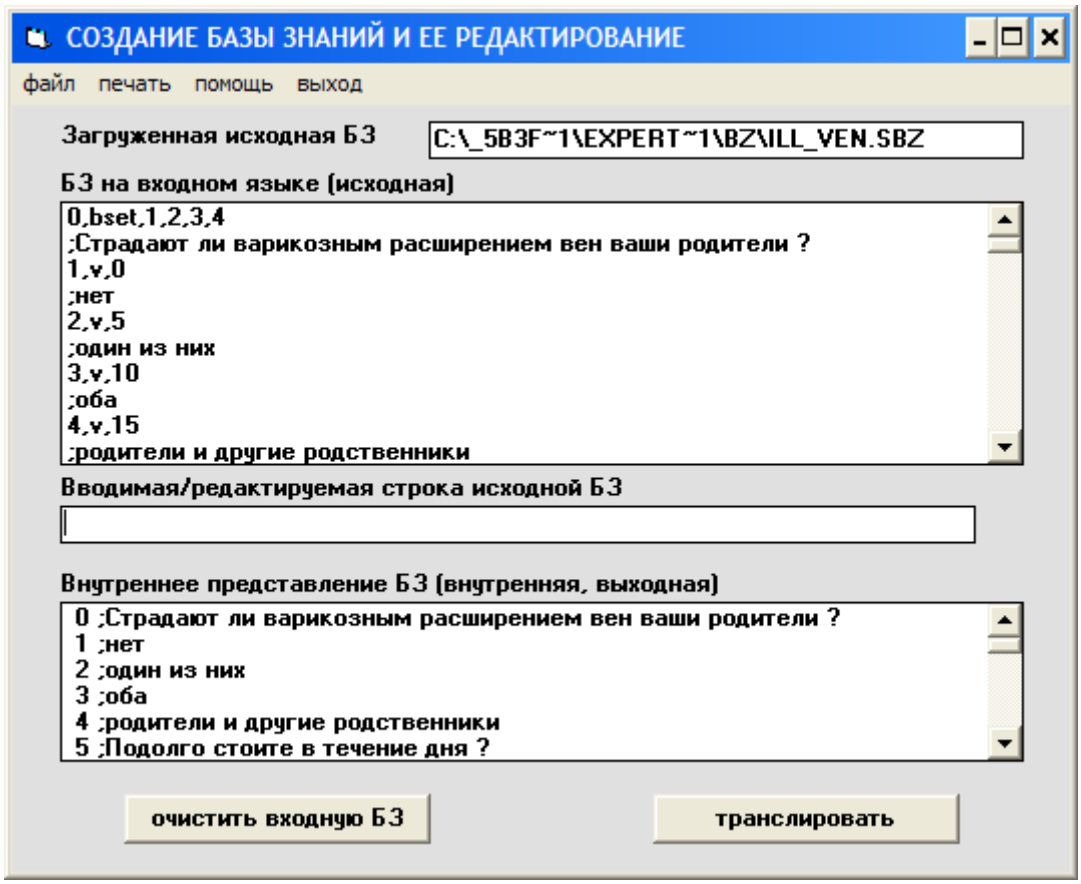


Рис. 10. Форма подсистемы ввода и редактирования БЗ

3. Диалоговые панели: с очередным вопросом – узел q (рис. 11), со списком альтернатив – узлы set, bset (рис. 12), с группой вопросов – узел jset (рис. 13).

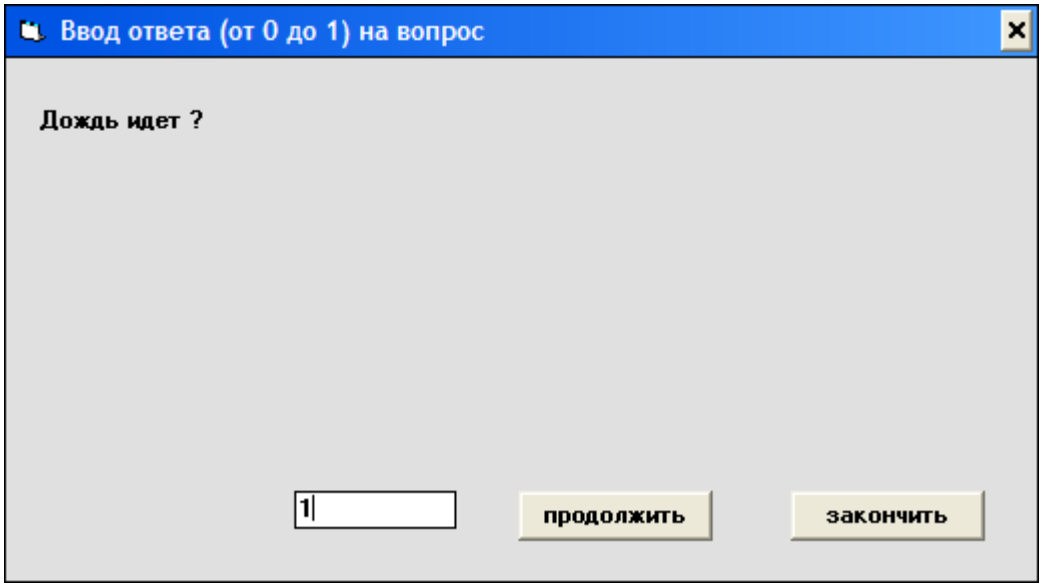


Рис.11. Диалоговая панель для ввода ответа

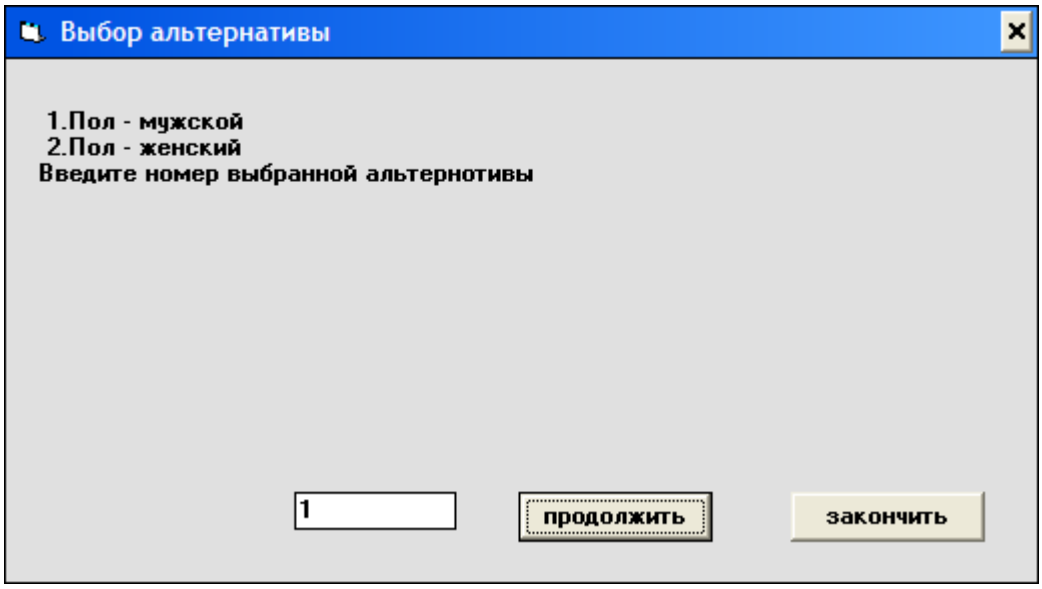


Рис. 12, Диалоговая панель для выбора альтернативного ответа

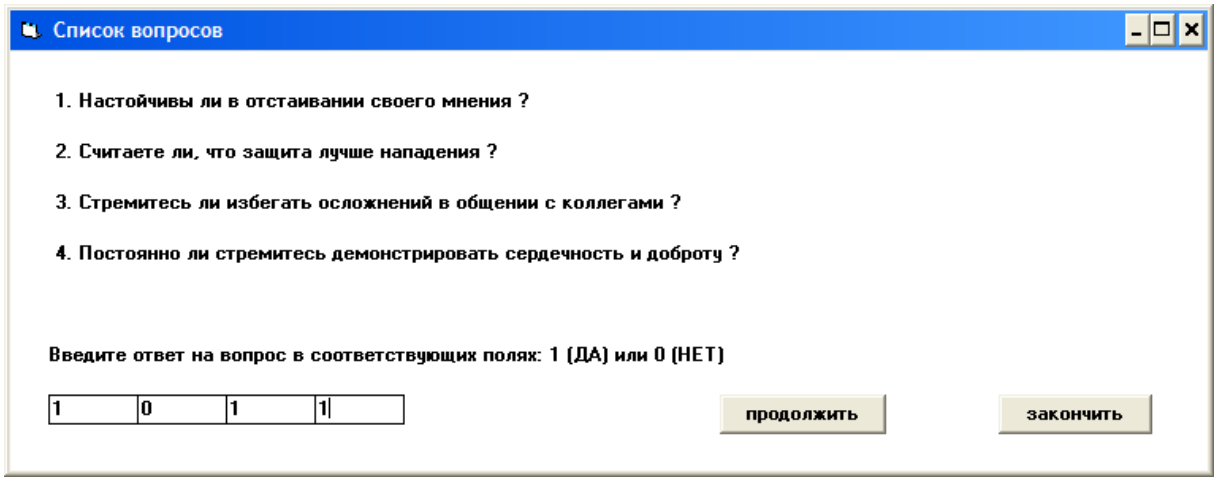


Рис. 13. Диалоговая панель для ввода группы ответов

4. На экране представлен результат вывода (рис. 14)

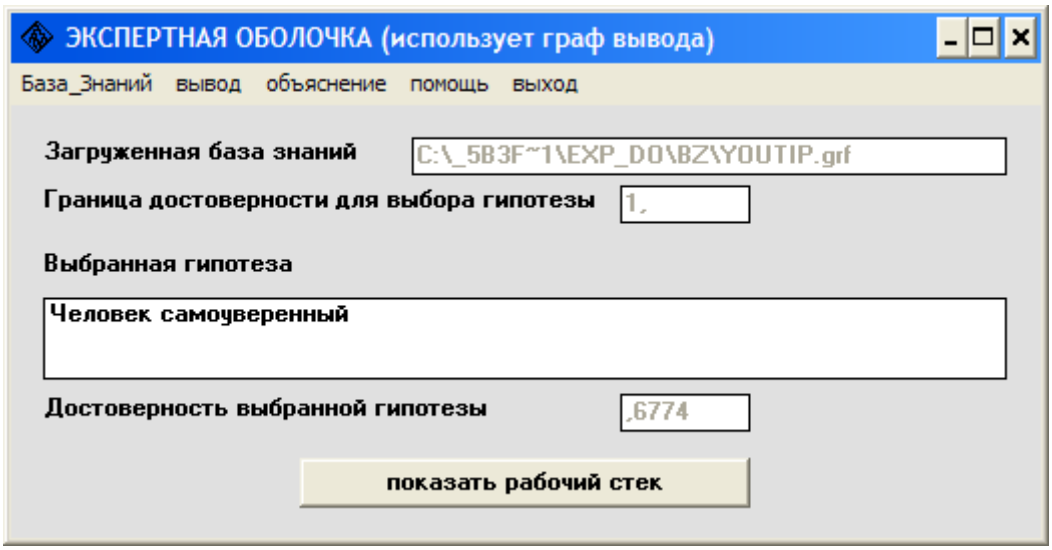


Рис. 14

5. Экран с объяснением, какие факты повлияли на выбор предложенной гипотезы (рис. 15)

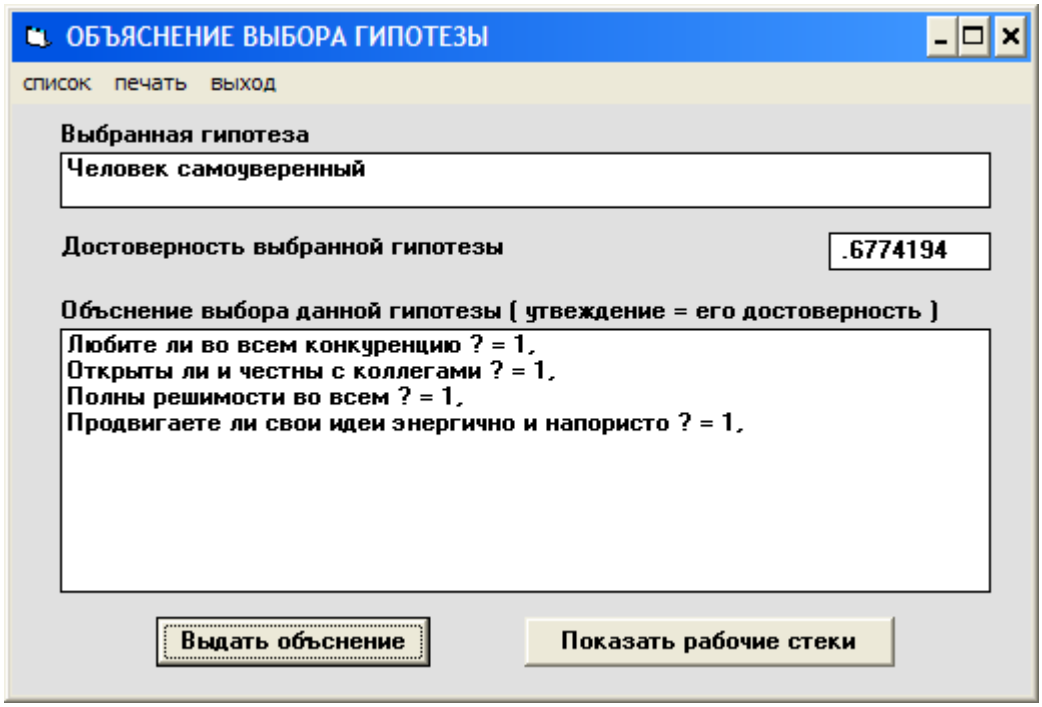


Рис. 15. Форма подсистемы объяснения

Здесь пункт меню «список» позволяет посмотреть ранжированный список всех конкурирующих гипотез (рис. 16)

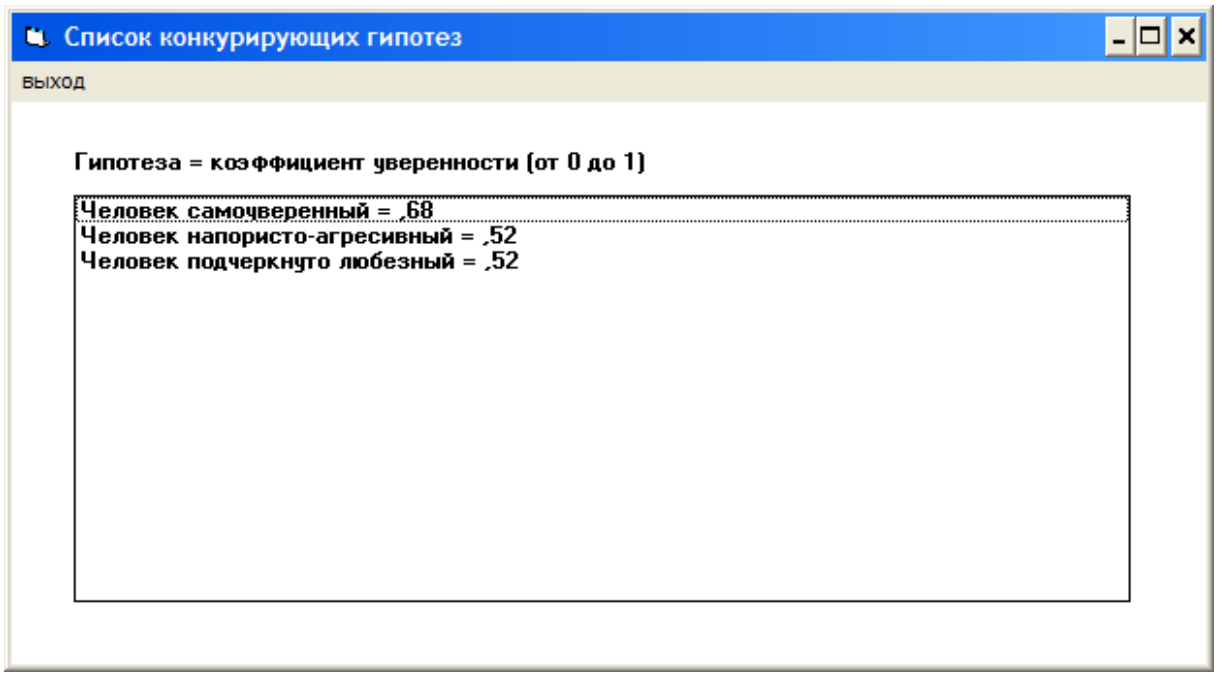


Рис.16. Форма со списком конкурирующих гипотез

Раздел 4. Экспертная система с продукционной моделью знаний

Рассматривается прототип экспертной оболочки, работающей с произвольной продукционной моделью знаний в виде набора правил типа

<КУ> **Если** <условие>, **то** <заключение>

Здесь <условие> - в общем случае совокупность условий, соединенных логической операцией «И» с возможным логическим префиксом «НЕ»; КУ – коэффициент уверенности (надежности, ценности) данного правила в процентах (<=100); заключение – утверждение, следующее из данного правила. Условие может быть фактом, требующим подтверждения пользователя (ввод числа в диапазоне 0 (твердое НЕТ) - 100 (твердое ДА)) или заключением промежуточного правила. В соответствии с этим, правила делятся на конечные (их заключения не встречаются в условиях других правил) и промежуточные.

Коэффициент уверенности заключения (число от 0 до 100) вычисляется согласно правилам нечеткой логики[2]

КУ(заключения) = КУ(правила)*0.01*min КУ(условий),

причем, если условию предшествует логическая операция НЕ, то КУ(условия) = 100 – КУ(условия).

База знаний (БЗ) должна обладать следующими свойствами:

- не противоречивостью - любому конкретному сочетанию условий сопоставляется только одно заключений;
- полнотой – для каждого «условия-заключения» имеется по крайней мере одно правило нижнего уровня (промежуточное), в котором это условие присутствует в качестве заключения;

- состоятельностью – для любой комбинации условий выводится конечное заключение.

Ниже приведен простой пример такой базы знаний, которая советует «брать ли Вам с собой зонтик»:

<100> Если «идет дождь», то «зонтик брать»
<100> Если «сегодня будет дождь», то «зонтик брать»
<100> Если «давление падает» И «низкая сплошная облачность» И «вчера был дождь» И «ветер слабый», то «сегодня будет дождь»
<90> Если «низкая сплошная облачность» И «вчера был дождь» И «ветер слабый», то «сегодня будет дождь»
<70> Если «низкая сплошная облачность» И «вчера был дождь», то «сегодня будет дождь»
<70> Если «низкая сплошная облачность» И «ветер слабый», то «сегодня будет дождь»
<100> Если НЕ «сегодня будет дождь», то «зонтик не брать»

Здесь первое, второе и последнее правила являются конечными, а остальные – промежуточными.

Оболочка состоит из трех подсистем: подсистема ввода правил, подсистема вывода конечного заключения и подсистема объяснения. Последняя собирает и выдает пользователю всю цепочку правил, которая согласно ответам пользователя привела к выбору предложенного конечного заключения.

С первой подсистемой работает инженер знаний. Он вводит правила на входном языке, а подсистема транслирует их во внутреннее представление и сохраняет в виде двух файлов - текстового и двоичного файла с массивом правил. Одновременно эта подсистема проводит синтаксический анализ введенной информации (ссылки на не существующие текстовые записи) и тестирует введенную базу знаний на полноту и непротиворечивость.

Со второй и третьей подсистемами работает пользователь, желающий получить рекомендации от конкретной базы знаний.

Покажем порядок кодировки базы знаний на входном языке оболочки для приведенного выше примера.

1. Собираем в массиве текстов все уникальные тексты, присутствующие во всех правилах. Причем все «тексты-факты» должны заканчиваться знаком вопроса (?)

0, Идет дождь?
1, Зонтик брать
2, Сегодня будет дождь
3, Давление падает?
4, Низкая сплошная облачность?
5, Вчера был дождь?
6, Ветер слабый?
7, Зонтик не брать

2. Формируем массив правил. Первая строка - заключение правила: КУ(правила)>0, номер соответствующего текста заключения в текстовом массиве. Последующие строки - условия правила: первый элемент 0 или -1 (условие с логическим префиксом НЕ), номер соответствующего текста условия в текстовом массиве

100, 0 - правило 1
0, 1
100, 0 - правило 2
0, 2
100, 2 - правило 3
0, 3
0, 4
0, 5
0, 6
90, 2 - правило 4
0, 4
0, 5
0, 6
70, 2 - правило 5
0, 4
0, 5
70,2 - правило 6
0, 4
0, 6
100, 7 - правило 7
-1, 2

Алгоритм тестирования не противоречивости БЗ.

for цикл по правилам
определяем число условий в правиле
for цикл по последующим правилам
if (*число условий совпадает*) *then*
if (*условия правил совпадают*) *then*
"*дубликат правил по условиям*"
выход из циклов
end if
end if
end for
end for

Алгоритм проверки полноты БЗ.

```
for цикл по правилам
   for цикл по условиям текущего правила
      if (условие правила без "?")
         fl = 0
         for цикл по правилам
            if (тексты условия и заключения совпадают) then
               fl = 1
               выход из цикла
            end if
         end for
         if (fl = 0) then
            "БЗ не полная"
            выход из циклов
         end if
      end if
   end for
end for
```

На рис.14 приведен экран первой подсистемы.

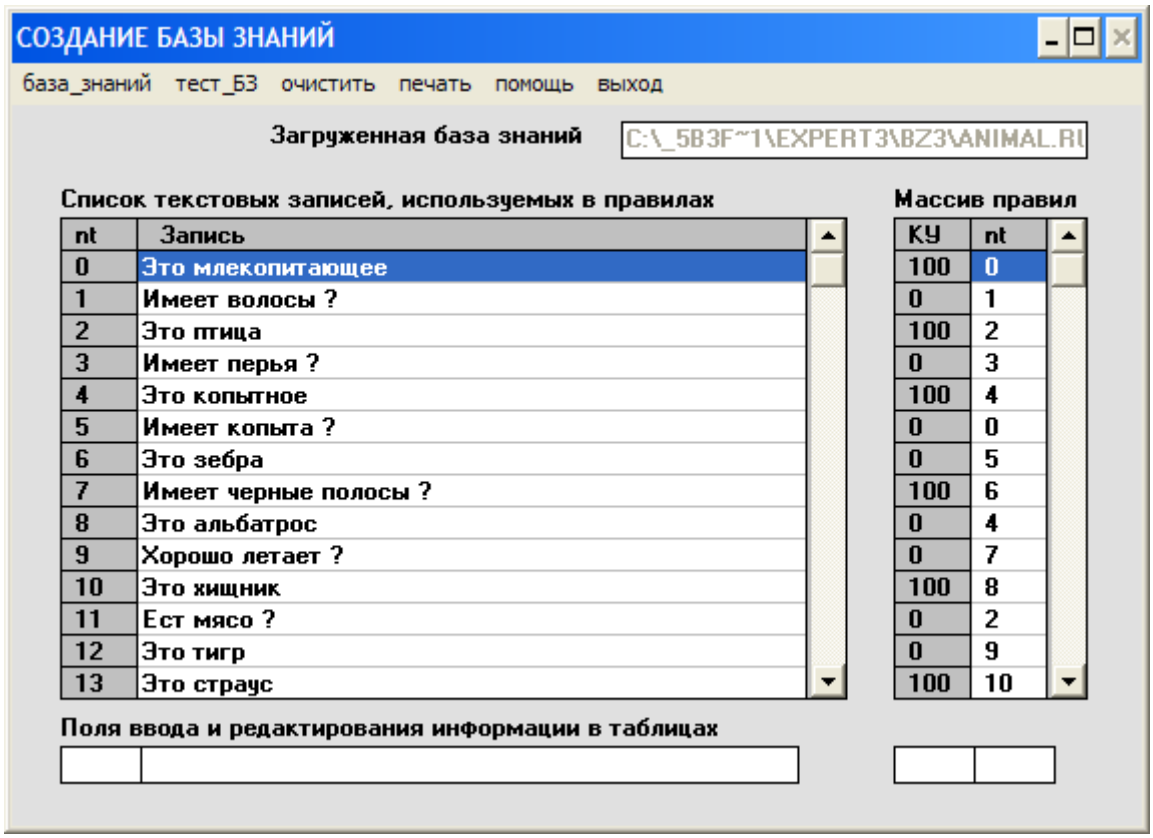


Рис. 14. Форма подсистемы ввода и редактирования БЗ

Подсистема вывода реализована в трех вариантах: с прямой, обратной и смешанной цепочками рассуждений.

В первом случае пользователю предъявляются сразу весь список фактов БЗ. Далее вывод идет от подтвержденных фактов (ответ пользователя >=60) и выбранных на основании этих фактов истинных промежуточных «условий-заключений» к конечному заключению. Алгоритм этой стратегии вывода дан ниже.
Вид рабочего массива правил, использованный в программе вывода

КУ	nt	L
--	--	--

где КУ>0 – КУ(правила), идентификатор начала правила – его заключение; КУ=0 – идентификатор условия (при этой стратегии логический префикс НЕ не используется); nt – номер текста, связанного с данным заключением или условием; L - вычисленный или полученный от пользователя коэффициент уверенности заключения или условия. L=-1 означает, что правило или условие еще не проанализированы.

1. Устанавливаем во всем массиве правил L= -1.
2. Цикл по списку фактов, выбранных пользователем
   2.1. Выбираем очередной факт
   2.2. Цикл по массиву правил
      Если это условие (КУ=0) и его nt = nt факта,
      то L условия = КУ факта
   2.2. Конец цикла
2. Конец цикла
3. Вызов п/п тестирования правил в массиве правил на истинность (L>50) и постановка истинных заключений в список выбранных заключений.
   Устанавливает fl=0 или 1(найдено хотя бы одно истинное правило).
4. Если не найдено ни одного истинного правила (fl=0), то выдаем соответствующее сообщение и заканчиваем вывод.
5. Пока fl=1, делай (- последующие проходы по списку выбранных заключений)
   5. 1. Цикл по списку выведенных заключений
      5. 1.1. Выбираем очередное заключение
      5..1.2. Цикл по массиву правил
         Если это условие (КУ=0) и его nt = nt заключения,
         то L условия = L заключения
      5.1.2. Конец цикла
   5.1. Конец цикла
   5.2. Вызов п/п тестирования правил в массиве правил на истинность (L>50) и постановка истинных заключений в список выбранных заключений или их корректировка, если они уже там присутствуют, но с более низким значением L. Если внесено изменение в список, то п/п возвращает fl=1.
5. Конец Пока

Алгоритм п/п тестирования правил. Здесь nr – размерность массива правил.

1. fl=0, ir=0 – номер первой строки обрабатываемого правила в массиве правил, Lmin=100
2. Цикл по массиву правил (i=1, nr-1 - пропускаем первое заключение)
   2.1. Если (КУ=0 и L<Lmin), то Lmin=L
   2.2. Если (КУ>0 или i=nr-1) (- конец очередного правила), то
      2.2.1. Если Lmin>50 (т.е. очередное правило истинно), то
         L заключения = КУ правила * Lmin * 0,01
      2.2.2. fl_yes=0

2.2.3. Цикл по списку выведенных заключений
(проверяем, есть ли уже это заключение в списке выведенных заключений)
Если (nt очередного заключения в списке =
nt анализируемого заключения И L старое < L новое), то
(L заключения = L новое, fl_yes=1, fl=1, выход из цикла)
2.2.3. Конец цикла
2.2.4. Если fl_yes=0, то (помещаем заключение в таблицу выведенных заключений, fl=1)
2.2.5. ir=i, Lmin=100
2.2. Конец если
2. Конец цикла

На рис.15 показано окно программы, реализующую данную стратегию вывода.

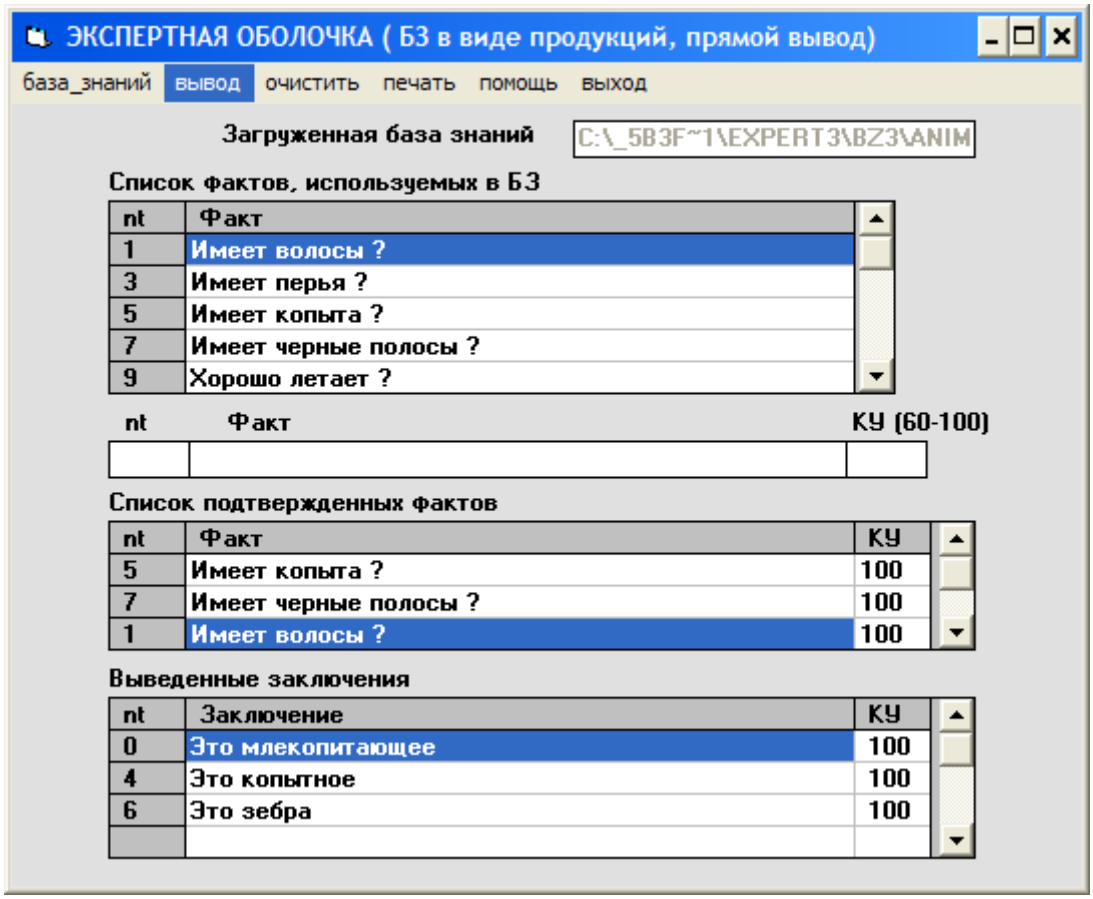
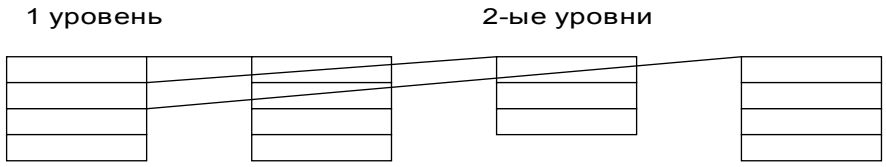


Рис. 15. Форма подсистемы вывода

Здесь форма представления промежуточной и конечной информации не требует дополнительной системы объяснения.

Недостаток прямой стратегии вывода в подобной реализации – для большой БЗ список предъявляемых фактов может быть слишком большим. В этом случае выдачу фактов можно сделать многоуровневой, введя для фактов дополнительный маркер уровня.



Естественно, это потребует некоторой доработки алгоритма вывода.

Во втором случае (обратная стратегия вывода) рассуждения направляются от конкретного конечного заключения к проверке истинности всех связанных с ним фактов.

Приведем алгоритм данной стратегии вывода.

Здесь используется следующий рабочий массив rmas(nr,4) для хранения правил

КУ	nt	L	fl
--	--	--	--

где nr – размерность массива; КУ>0 – КУ правила, идентификатор начала правила – его заключение; КУ=0 или -1 – идентификатор условия (КУ=-1 – перед условием стоит логический префикс НЕ); nt – номер текста, связанного с данным заключением или условием; L - вычисленный или полученный от пользователя коэффициент уверенности заключения или условия; fl=1 – означает: для заключения – это конечное правило, для условия – это факт, иначе fl=0.

Значение fl устанавливается при загрузке БЗ из файла по следующему алгоритму:

1. Цикл по массиву правил: i=0,nr-1
   1.2. Если это заключение (КУ>0), то
   fl=1
   Цикл по массиву правил: j=0,nr-1
   Если это условие (КУ<=0) И его nt совпадает с nt заключения, то fl=0 и выход из цикла
   Конец цикла
   rmas(i,3)=fl
   1.2. иначе (это условие)
   fl=0
   Если его текст содержит «?», то fl=1
   rmas(i,3)=fl
   1.2. Конец если
1. Конец цикла

Алгоритм обратной стратегии вывода, поиск «в ширину»:

1. Устанавливаем во всем массиве правил L= -1.
2. Цикл по массиву правил
   2.1. Загружаем в рабочий стек номер строки заключения очередного конечного правила (КУ>0, fl=1)
   2.2. Пока sp>=0, делай (sp - указатель рабочего стека)
   Выгружаем из стека номер строки заключения правила
   Lmin=100
   2.2.1. Цикл по строкам массива правил, начиная со следующей строки (цикл по условиям правила)
   2.2.1.1 Если это условие- не обработанный факт (КУ<=0, L=- 1, fl=1), то
   - выдаем его пользователю

- получаем ответ (от 0 до 100)
- ставим его в таблицу выданных фактов
- корректируем L этого факта во всех правилах массива, где оно присутствует как условие

2.2.1.1. Конец если

2.2.1.2. Если это условие с L=0, то

- Lзаключения правила = 0
- Если это промежуточное правило, то корректируем L этого заключения во всех правилах массива, где оно присутствует как условие
- убираем номер строки заключения этого правило из стека

2.2.1.2. Конец если

2.2.1.3. Если это не обработанное условие-заключение, то загружаем в стек номера строк заключений всех промежуточных правил, где оно присутствует в качестве заключения, и выходим из цикла 2.2.1

2.2.1.4. Если это условие И Lусловия < Lmin, то Lmin = Lусловия

2.2.1.5. Если это конец текущего правила (КУ>0 – начало очередного правила или это последняя строка массива), то

- Lзаключения = 0,01*КУправила*Lmin
- Если это промежуточное правило, то корректируем L этого заключения во всех правилах массива, где оно присутствует как условие
- Если Lзаключения > 50, то ставим его в таблицу выведенных заключений
- убираем номер строки заключения этого правила из стека
- выходим из цикла 2.2.1

2.2.1.5. Конец если

2.2.1. Конец цикла

2.2. Конец пока

2.3. Если найдено конечное заключение с Lзаключения >= предельное значение истинности, то вывод прекращаем

2. Конец цикла

3 Выдаем сообщение: «Не найдено истинного конечного заключения»

На рис.16, 17 приведены экран подсистемы обратного вывода и панель, с помощью которой пользователю предъявляются факты для анализа

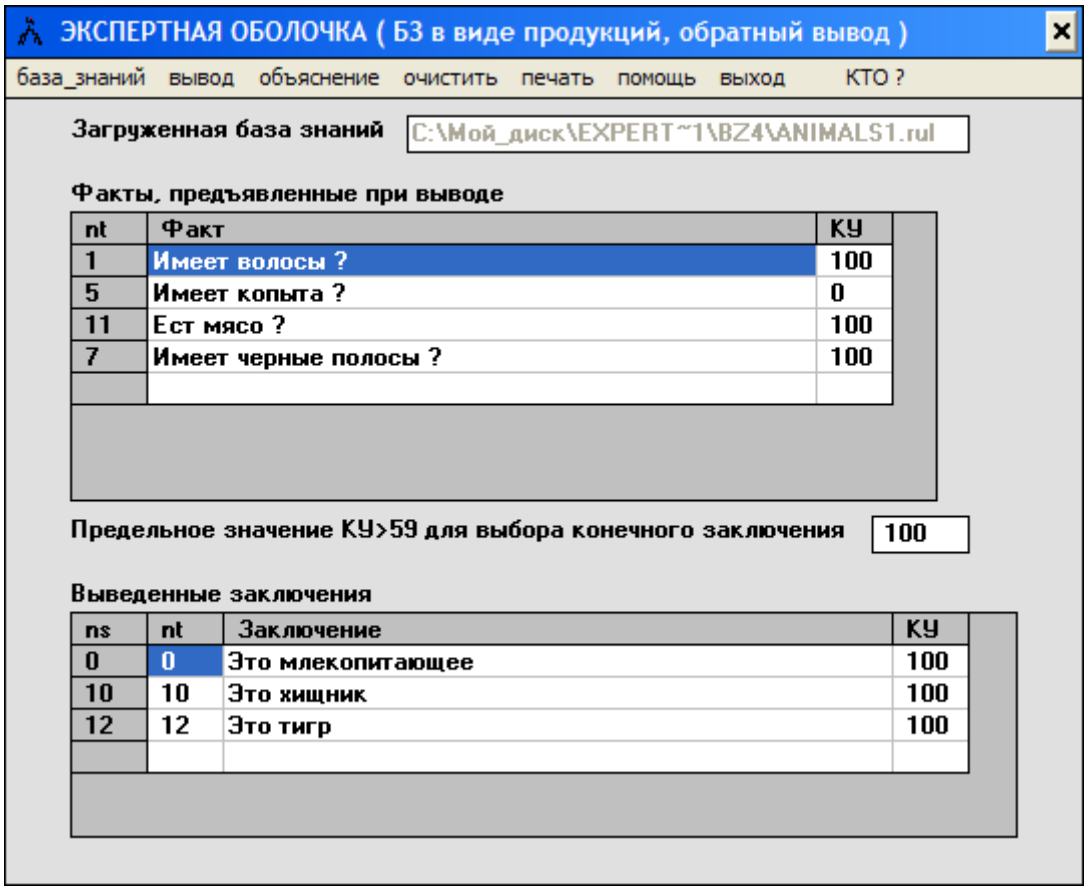


Рис. 16. Главная форма ЭС с обратной стратегией вывода

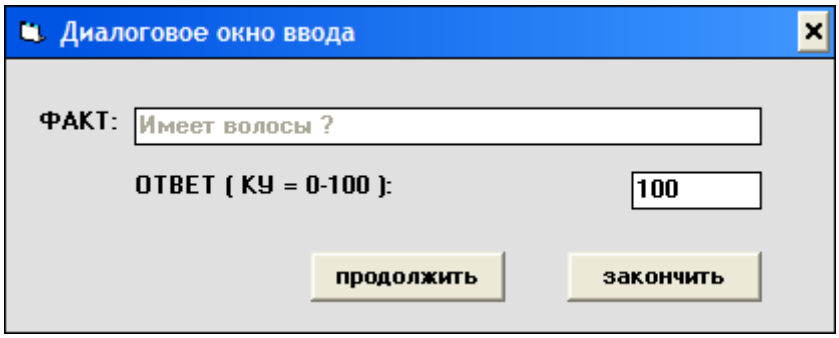


Рис. 17. Диалоговая панель для ввода ответа пользователя

В экспертной оболочке с обратной стратегией вывода предусмотрена подсистема объяснения. Она позволяет для каждого заключения, выбранного в списке выведенных заключений, построить цепочку правил, обеспечивших истинность этого заключения (см. верхнюю таблицу на рис. 18).

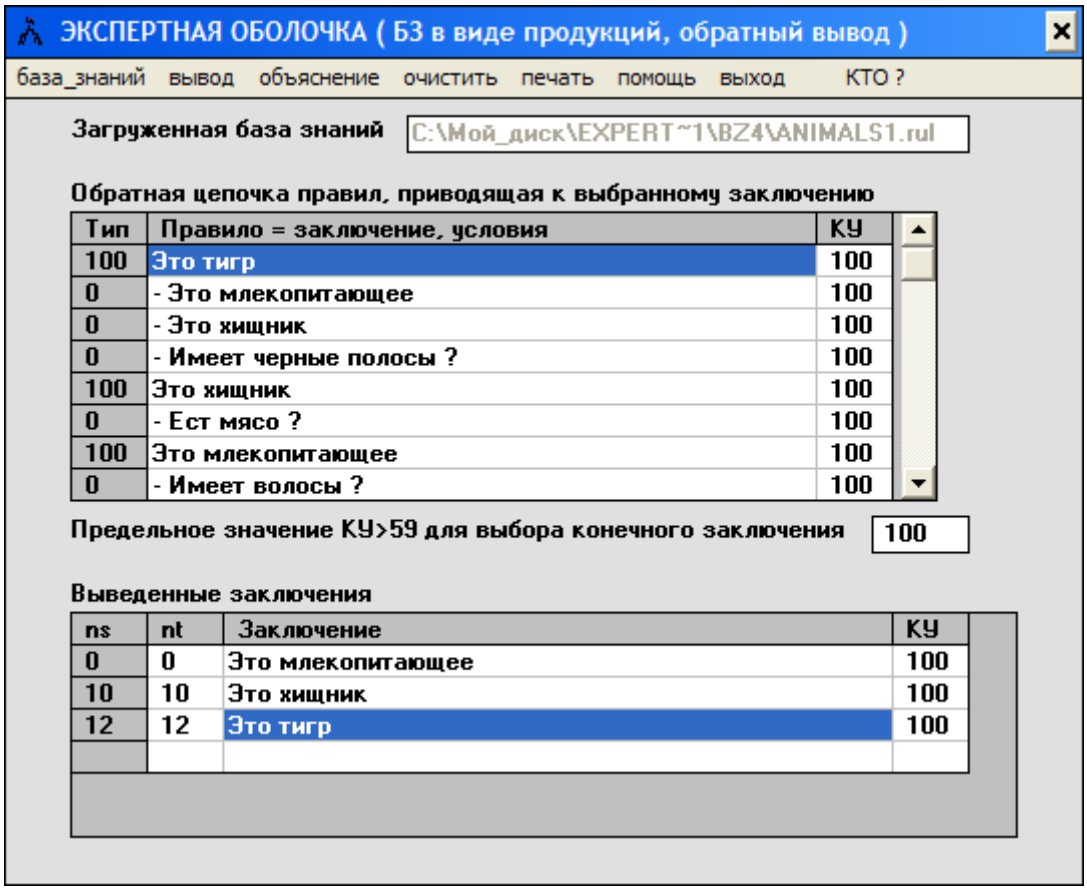


Рис. 18. Пример построения объяснения в виде прямой цепочки ипользованных правил

Алгоритм программы объяснения для любого выведенного заключения из таблицы выведенных заключений дан ниже.

1. Загружаем в рабочий стек номер строки в массиве правил того заключения, которое пользователь выбрал в таблице выведенных заключений
2. Пока sp>=0 (стек не пуст), делай
   2.1. Извлекаем из рабочего стека номер строки заключения очередного правила и помещаем его текст в формируемую таблицу обратной цепочки правил
   2.2. Цикл по условиям данного правила
   - помещаем текст условия в формируемую таблицу обратной цепочки правил
   - Если это условие не факт, то загружаем в рабочий стек порядковый номер строки заключения соответствующего промежуточного правила (nt заключения = nt условия И L заключения = L условия)
   2.2. Конец цикла
2. Конец пока

В третьем случае (смешанный вывод) первоначально находятся частоты, с которыми не анализированные условия-факты встречаются во всех цепочках правил, связанных с еще не обработанными на данный момент конечными правилами. После этого пользователю предъявляются все наиболее информативные условия-факты, частота которых попадает в некоторый ограниченный (задается самим пользователем) интервал частот, построенный возле максимальной частоты ((fmax – df) < f <fmax; fmax – максимальная частота, df – величина интервала =0,1,2,..).

После ответа пользователя корректируются условия в правилах, помечаются обработанные правила и снова пересчитываются частоты появления фактов в оставшихся цепочках конечных правил. Этот процесс повторяется до получения истинного конечного заключения.

Для хранения правил используется рабочий массив следующего формата,

КУ	nt	L	fl	fl_end
- -	- -	- -	- -	- -

где КУ>0 – КУ правила, идентификатор начала правила – его заключение; КУ= 0 или -1 – идентификатор условия (КУ= -1 – перед условием стоит логический префикс НЕ); nt – номер текста, связанного с данным заключением или условием; L - вычисленный или полученный от пользователя коэффициент уверенности заключения или условия; fl= 1 – означает: для заключения – это конечное правило, для условия – это факт; fl_end = -1 или 0 – правило или условие, соответственно, не обработано или его обработка уже закончена. Для правила последнее означает, что все факты, связанные с ним, уже предъявлены или его L=0.

Частоты хранятся в рабочем массиве структур с текстовыми записями БЗ следующего вида

Текстовая запись	частота
- -	- -

Приведем алгоритм подпрограммы, подсчитывающей для фактов их частоты.

0. Обнуляем ячейки буфера для хранения частот
1. Цикл по правилам
   1.1. Загружаем номер строки очередного не обработанного конечного правила в рабочий стек
   fl_inc=0 (частоты фактов для данного конечного правила еще не корректировались)
   1.2. Пока стек не пуст, делай
      1.2.1. Выбираем из рабочего стека номер строки очередного правила
      1.2.2. Цикл по его условиям
         - Если это не обработанный факт, то прибавляем к его частоте единичку; fl_inc=1 (имеет место корректировка факта)
         - Если это не обработанное условие-заключение, то загружаем в стек номера строк всех не обработанных промежуточных правил, где оно присутствует в качестве заключения
      1.2.2. Конец цикла
   1.2. Конец пока
   1.3. Если fl_inc=0 (ни одна частота не корректировалась), то
      - для конечного правила fl_end=0 – обработка правила закончена, поскольку коэффициент уверенности его заключения (L) уже не будет корректироваться дополнительными фактами (все промежуточные правила для его условий уже обработаны)
      - Если L заключения>=КУ предельное (устанавливается пользователем), то fl_exit=1 – получено истинное конечное заключение; выход из подпрограммы.
   1.3. Конец если
1. Конец цикла

На рис.19 и 20 показаны экраны программы на стадиях вывода и объяснения (последнее обрабатывается по тому же алгоритму, что и случае обратного вывода). Вывод инициируется выбором пункта меню «**вывод**», а каждая новая партия фактов выбирается кнопкой «**продолжить**» до получения истинного конечного заключения.

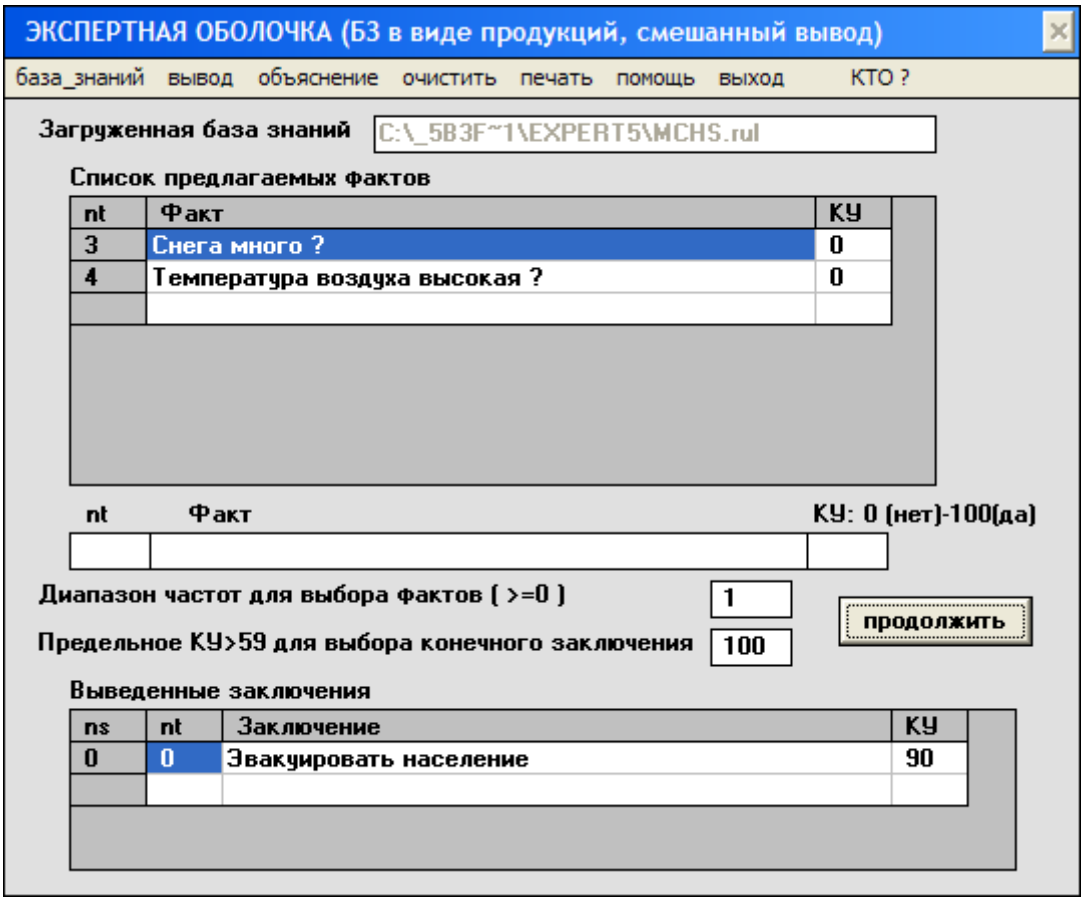


Рис. 19. Главная форма ЭС со смешанной стратегией вывода

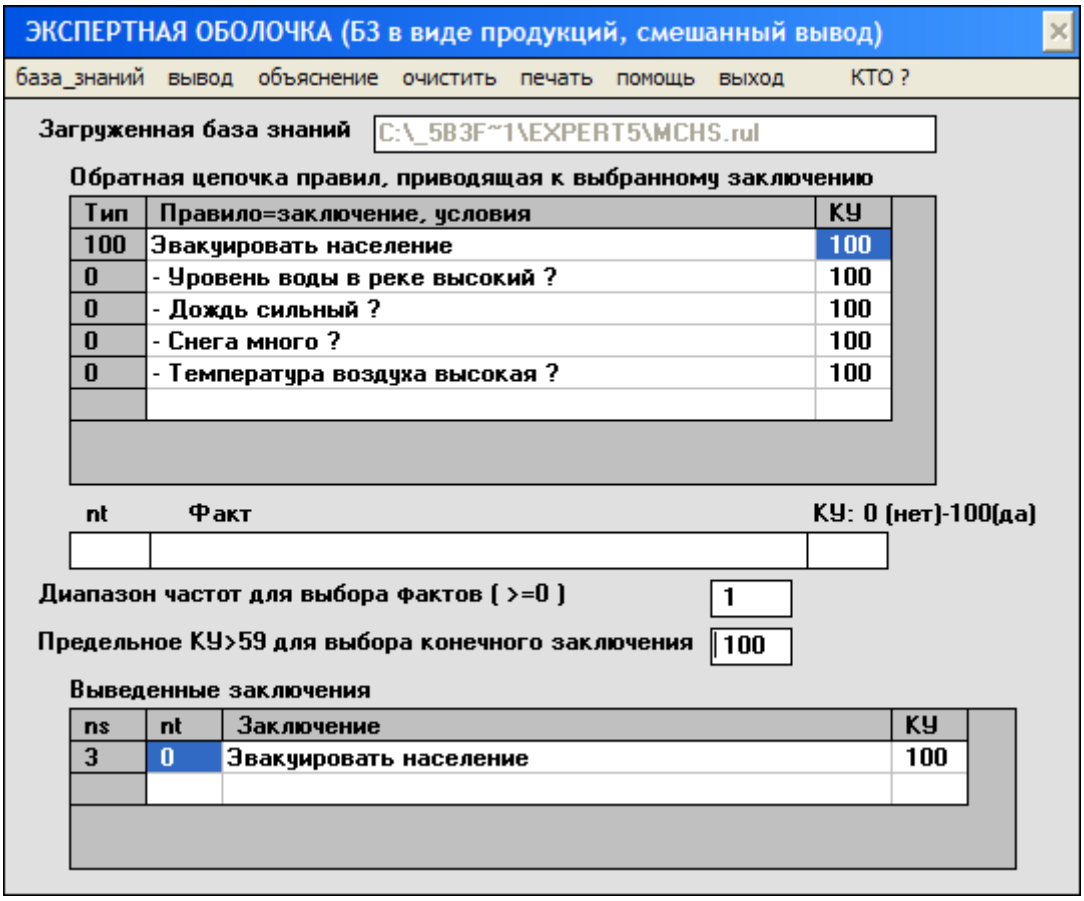


Рис. 20. Пример построения прямой цепочки использованных при выводе правил

Ниже приведены алгоритмы подпрограмм, обрабатывающих выбор этих элементов управления.

Алгоритм подпрограммы обработки пункта меню «**вывод**»

1. Считываем значения предельного КУ и диапазон частот для выбора фактов
2. Инициализируем компоненты в массиве правил: L=-1, fl_end=-1
3. Устанавливаем fl_exit=-1 (еще не найдено ни одного истинного конечного заключение)
4. Вызов подпрограммы, которая подсчитывает частоты появления фактов во всех цепочках правил, связанных с конечными правилами БЗ
5. Выбор фактов, частота которых попадает в установленный диапазон частот, и постановка их в таблицу «Список предлагаемых фактов» для анализа пользователю.

Алгоритм подпрограммы обработки кнопки «**продолжить**»

1. Цикл по списку выданных фактов
   1.1. Выбираем очередной факт
   1.2. Цикл по массиву правил
   Если текст условия правила совпадает с текстом выбранного факта, то корректируем с учетом возможного логического префикса НЕ значение его L – коэффициент истинности/уверенности
   1.2. Конец цикла

1.3. Вызываем подпрограмму анализа правил на их истинность, которая возвращает:
FL=0 – если корректировалось L хотя бы одного заключения, иначе FL=-1;
fl_exit=1 – если найдено истинное конечное заключение, иначе fl_exit=-1
1.4. Если fl_exit=1 (найдено конечное истинное заключение), то
выдаем его текст пользователю и заканчиваем вывод
1. Конец цикла
2. Пока FL=0 делай (последующие проходы по правилам)
2.1. Цикл по массиву правил
Если это обработанное промежуточное правило, то корректируем условия правил, тексты которых совпадают с текстом его заключения (если только Lусловия<Lзаключения данного промежуточного правила; а для условия с префиксом НЕ - если только Lусловия>Lзаключения)
2.1. Конец цикла
2.2. Вызываем подпрограмму анализа правил на их истинность
2.3. Если fl_exit=1 (найдено конечное истинное заключение), то
выдаем его текст пользователю и заканчиваем вывод
2. Конец пока
3. Вызов подпрограммы, которая подсчитывает частоты появления фактов во всех цепочках правил, связанных с конечными правилами БЗ
4. Если fl_exit=1 (найдено конечное истинное заключение), то
выдаем его текст пользователю и заканчиваем вывод
5. Если fl_exit=-1 (не найдено конечное истинное заключение) И не найдено ни одного нового факта, то выдаем сообщение: «Не найдено истинного конечного заключения» и заканчиваем вывод
6. Выбор фактов, частота которых попадает в установленный диапазон частот, и постановка их в таблицу «Список предлагаемых фактов» для анализа пользователю.

Алгоритм подпрограммы анализа правил на их истинность

1. Цикл по массиву правил
1.1. Находим начало очередного правила (КУ>0)
1.2. Lmin=100, fl_zerro=0, fl_zakl=0, FL=-1
1.3. Цикл по его условиям
- Если условие есть факт с L=0, то fl_zerro=1
- Если условие есть заключение промежуточного правила, то fl_zakl=1
- Если Lусловия<Lmin, то Lmin=Lусловия
1.3. Конец цикла
1.4. Если fl_zerro=1, то Lзаключения=0, Lmin=0
1.5. Lтекущее=Lmin * КУправила * 0,01
1.6. Если Lтекущее=100,то fl_end=0 – обработка правила закончена
1.7. Если fl_zakl=0 (нет условий-заключений) И Lтекущее>=0, то fl_end=0 – обработка данного правила закончена
1.8. Если Lзаключения<Lтекущее, то Lзаключения=Lтекущее – корректируем коэффициент истинности заключения данного правила; FL=0- произошла корректировка
1.9. Если Lзаключения корректировалось И оно >50, то ставим его в таблицу выведенных заключений
1.10. Если данное правило конечное (его fl=1) И Lего заключения>=КУпредельное И его fl_end=0 (обработка правила закончена), то fl_exit=1 – вывод закончен (получено истинное конечное заключение), выход из подпрограммы

1. Конец цикла

Раздел 5. Экспертная система с семантической моделью знаний

Рассматривается экспертная оболочка, работающая с моделью знаний в виде семантической сети[1,2.3] с произвольными, бинарными дугами-отношениями между двумя узлами сущностями или между сущностью и значением отношения-признака. В сети допускается наследование свойств для любых отношений. Для этого достаточно поставить символ «*» перед названием соответствующего отношения. Например,

«ястреб» - «*это» - «птица»
«ястреб» – «окрас» - «серый»
«птица» - «имеет» - «оперение»

Здесь сущность «ястреб» через отношение «*это» наследует свойства сущности «птица»; признаку «окрас» присвоено значение «серый».

Оболочка состоит из двух подсистем: подсистема ввода элементов сети и подсистема сбора информации.

С первой подсистемой работает инженер знаний. Он вводит семантическую сеть на входном языке, а подсистема сохраняет ее в виде двух файлов – текстового, состоящего из названий сущностей, отношений, отношений-признаков и их значений, и двоичного файла с массивом номеров текстовых записей следующих связок сети

«левая сущность - отношение/признак - правая сущность/значение»

Одновременно эта подсистема осуществляет проверку правильности ссылок из второго файла на записи первого.

С подсистемой вывода работает пользователь, желающий получить информацию, хранящейся в конкретной базе знаний.

Подсистема позволяет собирать информацию о сущностях либо для одного выбранного типа отношения/признака, либо для всех типов сразу. При этом можно включать или отключать признак наследования. Помимо этого подсистема может выбирать все экземпляры, принадлежащие выбранной сущности-классу и связанные с ней через отношения с признаком наследования.

Экспертная система имеет простой понятный интерфейс на этапах ввода базы знаний (рис. 21) и вывода экспертной информации (рис. 22), а ее база знаний легко пополняется и модифицируется.

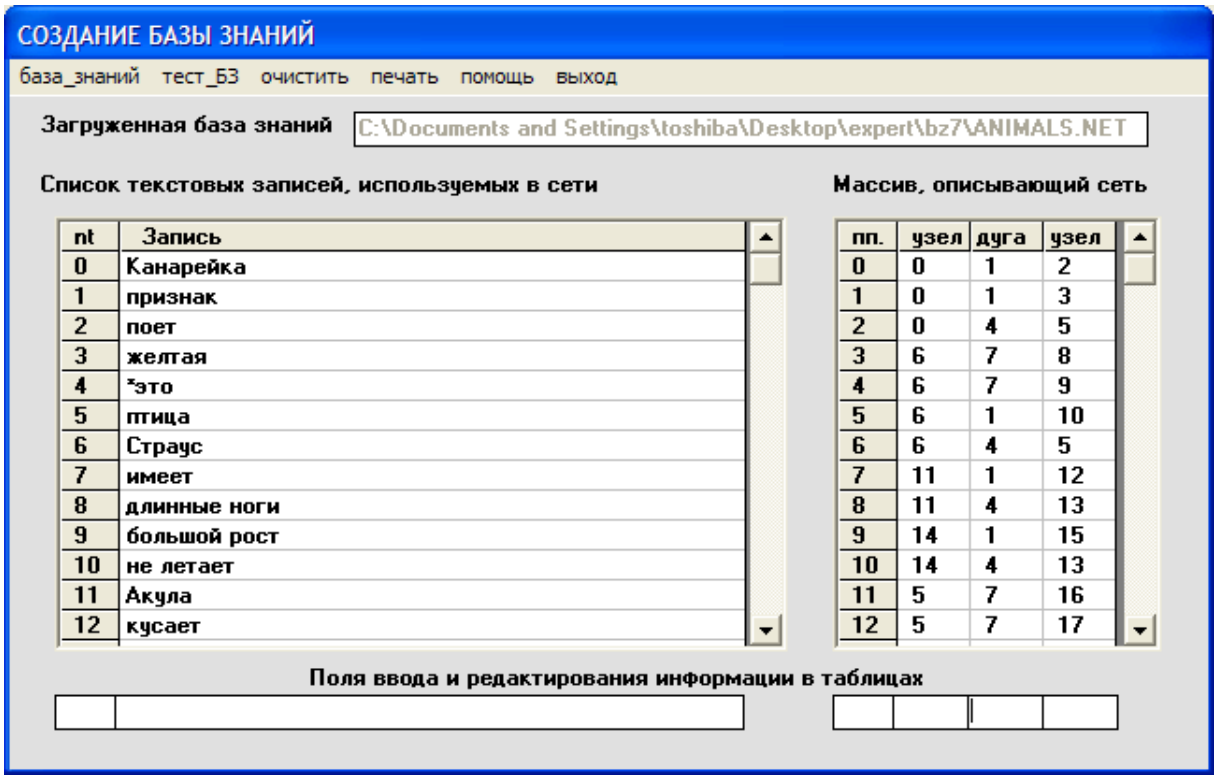


Рис. 21. Форма для ввода семантической БЗ

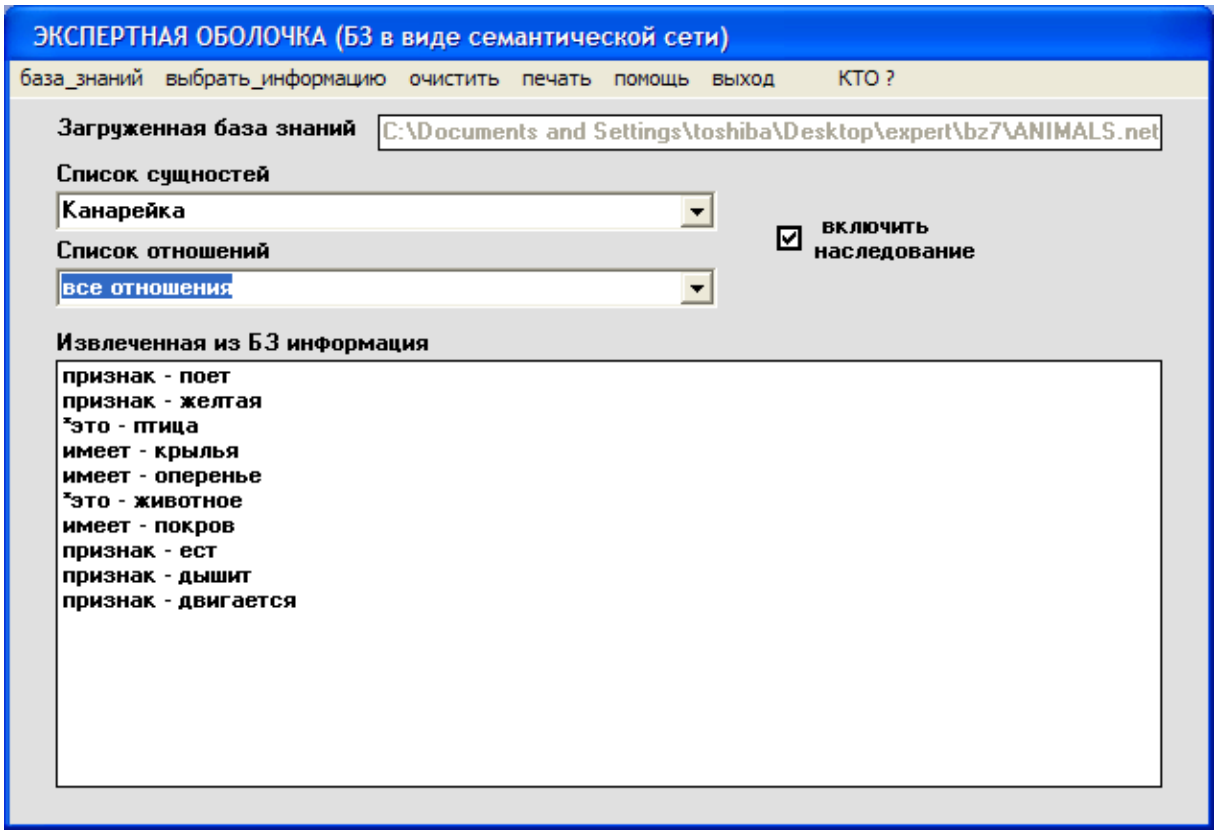


Рис.22. Главная форма ЭС с извлеченной из БЗ информацией

На рисунках представлены результаты работы следующей тестовой семантической БЗ: (отношения с наследованием следует ставить последними в перечислении отношений для конкретной сущности)

Канарейка – признак – поет
- признак – желтая
- *это – птица
Страус – имеет – длинные ноги
- имеет – большой рост
- признак – не летает
- *это – птица
Акула – признак – опасна
- признак – быстро плавает
- *это – рыба
Лосось – имеет – розовое мясо
- признак – съедобен
- признак – на нерест плывет вверх по реке
- *это – рыба
Птица – имеет – оперение
- имеет – крылья
- *это – животное
Рыба – имеет – плавники
- имеет – жабры
- признак – живет в воде
- *это – животное
Животное – имеет – покров
- признак – ест
- признак – двигается
- признак - дышит

Ниже приведен текст подпрограммы на языке Visual Basic, осуществляющей сбор затребованной информации (для этого вначале выбираются тексты в списках сущностей и отношений, а затем пункт меню «**выбрать_информацию**»). Здесь rmas(.,.) – массив для хранения сети; nr – размерность этого массива; zap(.) – рабочий массив двухкомпонентных структур вида: zap(.).t – текст сущности, отношения или значения, zap(.).fl=0/1 – 1 – если это текст отношения с наследованием (инициируется при загрузке БЗ).

```
Private Sub resumer_Click()
Dim i%, s$
Dim sp%, fl_end%, wnt1%, n%, m%, wnt2%
ReDim stek(nr)        ‘запрос буфера под стек
  List1.Clear             ‘чистим список выдачи собранной информации
  ilist1 = -1
  If icombo1 = -1 Or icombo2 < 2 Then
  ‘если ниспадающие списки сущностей и отношений пусты, то выдаем сообщение
    MsgBox "БЗ не корректно составлена !", 48, "Сообщение"
    Exit Sub
  End If
  sp = -1
```

```
 fl_end = 1
 Select Case ind2        ‘выбор альтернативы
   Case icombo2 - 1   'выбрано -> выдать все экземпляры для выбранной
сущности-класса, т.е. собрать все левые сущности, для которых правые сущности
совпадают с сущностью-классом, выбранной из списка сущностей, и связанных с ней
через отношение с наследованием («*это». «*является» и т.д.)
      wnt1 = nt1                ‘номер текста сущности, выбранной в списке combo1
      For i = 0 To nr – 1       ‘цикл по массиву сети
       If rmas(i, 2) = wnt1 Then    'правая сущность = выбранной сущности
        n = rmas(i, 1)              'отношение
        If zap(n).fl = 1 Then       'наследует значения
         m = rmas(i, 0)             'номер текста экземпляра
         s = RTrim$(zap(m).t)       ‘s = текст экземпляра
         List1.AddItem s            'добавляем текст экземпляра в список выдачи
         ilist1 = ilist1 + 1
        End If
       End If
      Next i
   Case icombo2               'выбрано -> выдать все отношения для выбранной сущности,
т.е. собрать все пары: отношение/признак-правая сущность/значение признака, для
которых левая сущность совпадает с выбранной из списка сущностей и, если включено
наследование, добавить такие же пары, связанные с правой сущностью через
отношение с наследованием
       wnt1 = nt1                 'номер текста выбранной сущности
       Do While fl_end = 1        ‘делай пока fl_end=1
        For i = 0 To nr – 1       ‘цикл по массиву сети
         If rmas(i, 0) = wnt1 Then
         ‘если номера текущей и выбранной сущностей совпадают
          n = rmas(i, 1)          'номер текста текущего отношения
          If zap(n).fl = 1 And fl_nasled = 1 Then
          ‘если оно имеет признак наследования и наследование необходимо учитывать,
          то загружаем в стек порядковый номер в массиве сети сущности, свойства
          которой наследуются
            sp = sp + 1
            stek(sp) = i
          Else
            m = rmas(i, 2)        'номер текста значения текущего отношения
            ‘собираем в строке s тексты отношения и его значения
            s = RTrim$(zap(n).t) & " - " & RTrim$(zap(m).t)
            List1.AddItem s       'добавляем текст строки s в список выдачи
            ilist1 = ilist1 + 1
          End If
         End If
        Next i
        If sp = -1 Then   ‘если стек пуст
         fl_end = 0
        Else
        ‘иначе выбираем из стека номер строки с сущностью, свойства
        ‘которой наследуются
         i = stek(sp)
```

```
          sp = sp - 1
          wnt1 = rmas(i, 2)
          n = rmas(i, 1)
          m = wnt1
          s = RTrim$(zap(n).t) & " - " & RTrim$(zap(m).t)
          List1.AddItem s      'добавляем текст строки s в список выдачи
          ilist1 = ilist1 + 1
        End If
      Loop
  Case Else          'выбрано -> конкретное отношение, т.е. собрать все правые сущности/
значения, для которых левая сущность и отношение совпадают с выбранными из
списков сущностей и отношений, при этом просмотреть и цепочки передаваемые по
наследованию (если включено наследование)
      wnt1 = nt1                'номер текста выбранной сущности
      wnt2 = nt2                'номер текста выбранного отношения
      Do While fl_end = 1   ‘делай пока fl_end=1
       For i = 0 To nr – 1     ‘цикл по массиву сети
        If rmas(i, 0) = wnt1 Then
        ‘если номера текущей и выбранной сущностей совпадают
         n = rmas(i, 1)          'номер текста текущего отношения
         m = rmas(i, 2)          'номер текста его значения
         If n = wnt2 Then
         ‘если номера текущего и выбранного отношений совпадают
          s = RTrim$(zap(m).t)  ‘s = текст значения отношения
          List1.AddItem s    'добавляем текст строки s в список выдачи
          ilist1 = ilist1 + 1
         End If
         If zap(n).fl = 1 And fl_nasled = 1 Then
         ‘если отношение имеет признак наследования и наследование необходимо
         ‘учитывать
        ‘загружаем в стек порядковый номер в массиве сети сущности, свойства
        которой     ‘наследуются
          sp = sp + 1
          stek(sp) = m
         End If
        End If
       Next i
       If sp = -1 Then     ‘если стек пуст
        fl_end = 0
       Else
       ‘иначе выбираем из стека номер строки с сущностью, свойства которой
       ‘наследуются
        wnt1 = stek(sp)
        sp = sp - 1
       End If
      Loop
 End Select
 If ilist1 = -1 Then
  MsgBox "В БЗ требуемая информация отсутствует !", 48, "Сообщение"
 Else
```

```
  MsgBox "Требуемая информация выдана  !", 48, "Сообщение"
 End If
   End Sub
```

Дополнительно перечислим действия, которые выполняются оболочкой при выборе и загрузке конкретной БЗ:
1. Загрузка в рабочий массив rmas(.,.) из двоичного файла информации с описанием элементов сети.
2. Перенос из текстового файла в рабочий массив структур zap(.) уникальных текстов сети.
3. Инициализация в рабочем массиве структур zap(.) компонент zap(.).fl (1 – если это текст отношения с наследованием свойств, 0 – иначе).
4. Размещение в первом выпадающем списке (combo1) всех уникальных левых сущностей сети.
5. Размещение во втором выпадающем списке (combo2) всех уникальных отношений/признаков сети.
6. Добавление в конец второго выпадающего списка двух записей: «экземпляры/части» - выбор всех элементов, принадлежащих к выбранной левой сущности-классу; «все отношения» - сбор всех отношений для выбранной левой сущности.
7. Установка переключателя наследования в положение «включено».

Возможный путь доработки системы:

Можно сократить размеры списков сущностей и отношений, если в них собирать только специально помеченные тексты. Тогда при выдаче информации остальные сущности и отношения будут появляться автоматически за счет наследования, а также при запросе всех отношений или всех экземпляров.

Значение свойств сущностей сделать более приоритетными по сравнению с наследуемыми значениями.

Раздел 6. Модель экспертной системы мониторинга с использованием правил и фреймов

Система осуществляет мониторинг уровня воды на реке в нескольких городах, расположенных на ее берегу, и зависимости от прогнозируемой ситуации передает администрации города сообщение о рекомендуемых мероприятиях.

Используются экспертные знания в виде следующих 10-ти правил.

Если:

№	Уровень	Дождь	Снег	Температура	Ситуация (в верховье)
1	высокий	сильный	много	высокая	-
2	высокий	-	много	высокая	наводнение
3	высокий	сильный	-	-	наводнение

то наводнение. **Рекомендация для администрации города** – провести эвакуацию населения.

Если:

№	Уровень	Дождь	Снег	Температура	Ситуация (в верховье)
1	высокий	-	-	-	наводнение
2	средний	сильный	много	высокая	-
3	средний	сильный	мало	высокая	-
4	средний	умеренный	много	высокая	-
5	высокий	сильный	-	-	-
6	высокий	отсутствует	много	высокая	-

то возможно наводнение. **Рекомендация для администрации города** – подготовить план эвакуации населения.

Во всех остальных случаях - штатная ситуация. **Рекомендация для администрации города** – никаких специальных мероприятий не требуется.

Каждому городу сопоставлен фрейм-объект, 5 слотов которого описывают наблюдаемые параметры, а последний - прогнозируемую в соответствии с параметрами и приведенными выше правилами текущую ситуации на реке (см. рис. 23).

С фреймом связаны две процедуры.

Первая выполняется при изменении значения любого входного параметра фрейма. Ее задача - по текущим значениям параметров найти истинное правило и поместить соответствующий прогноз в последний слот фрейма. В процедуре правила реализованы в виде условных операторов.

Вторая процедура запускается при изменении значения слота – прогноза. Ее задача - передать сообщение администрации города о мероприятиях, которые необходимо провести в соответствии с текущим прогнозом ситуации на реке (см. рис. 24).

На рис. 25 приведена панель, используемая для передачи выбранных параметров наблюдения во фрейм-образ для заданного города. Значения параметров выбираются из ниспадающих фиксированных списков.

Главной панелью является панель, с помощью которой можно создавать и удалять фреймы-образцы (см. рис. 25).

Опишем порядок функционирования системы.

Первой загружается главная панель. Она инициирует и редактирует двумерный массив arr(6,n) для хранения значений слотов фреймов-образцов. Здесь n – текущее число сохраненных фреймов-образцов. Шаблон самого фрейма «зашит» в самой программе. Значения слотов хранятся в указанном массиве в виде индексов текстов, выбранных из фиксированных ниспадающих списков.

После завершения формирования списка объектов и нажатия кнопки «Загрузить панель наблюдения» загружаются все остальные панели.

Выбрав на рис. 4 объект, можно изменить или посмотреть ранее введенные параметры наблюдения для слотов соответствующего фрейма-образца.

При фиксации элемента каждого из 5-ти ниспадающих списков (на рис. 26) производится его автоматическая передача в соответствующее текстовое поле на панели с текущим фреймом (рис. 23). В свою очередь, изменение содержимого любого текстового поля (кроме последнего) приводит к запуску первой из описанных выше процедур, которая изменяет содержимое текстового поля последнего слота. При

изменении содержимого последнего запускается вторая процедура, которая выдает сообщение на панель рис. 24.

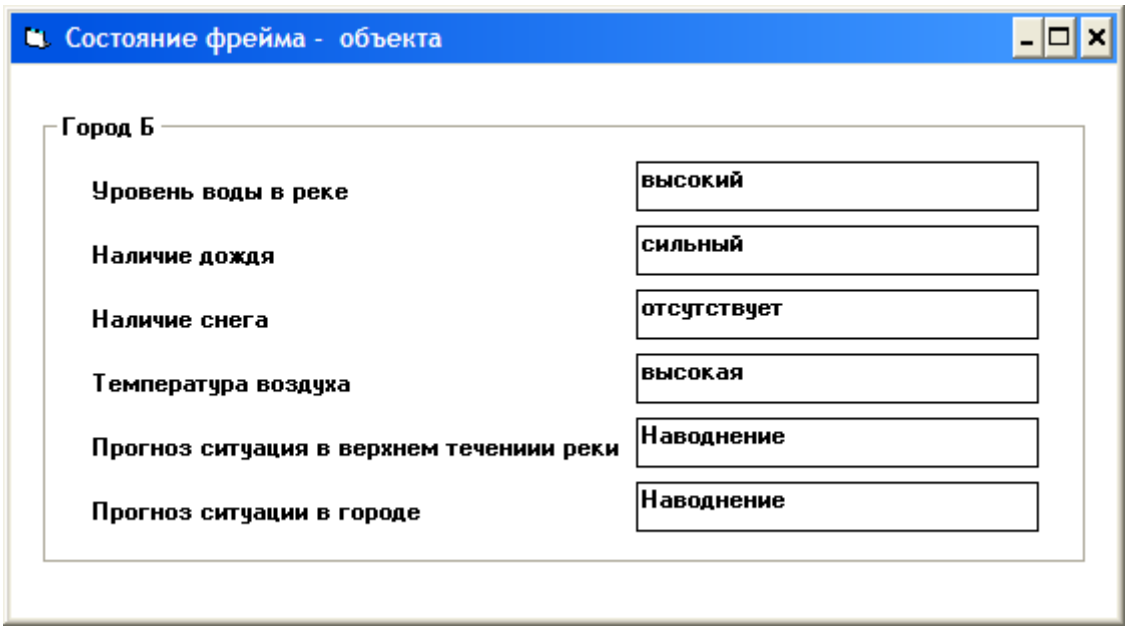


Рис. 23. Форма для отображения фрейма состояния города

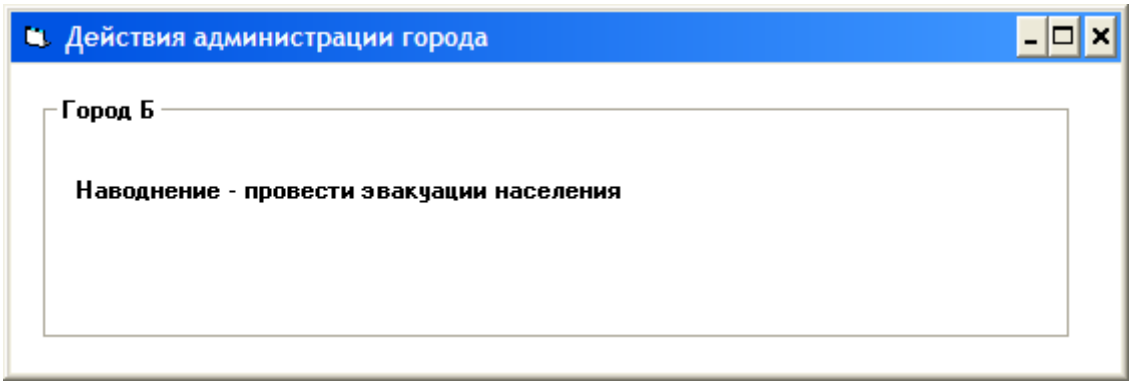


Рис. 24. Форма синформацией для администрации города

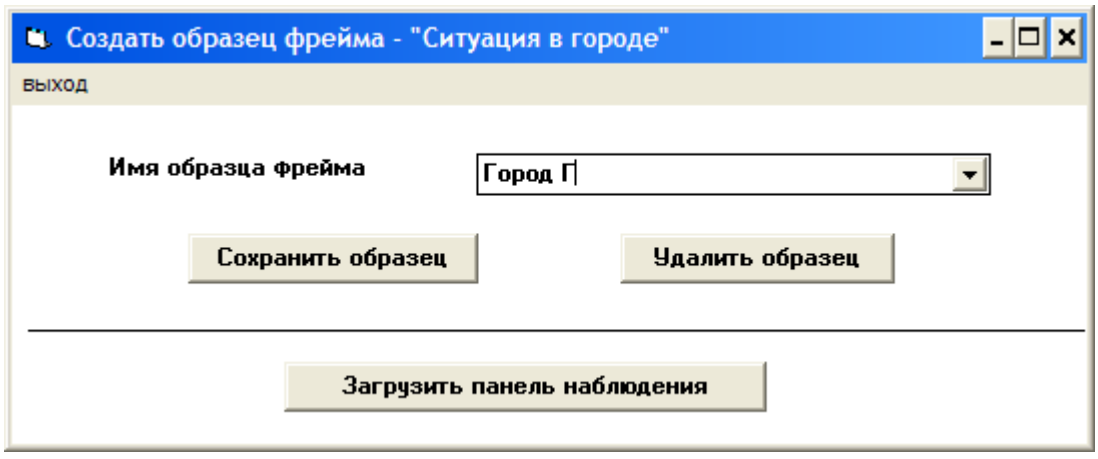


Рис. 25. Форма для создания и редактирования образцов фреймов

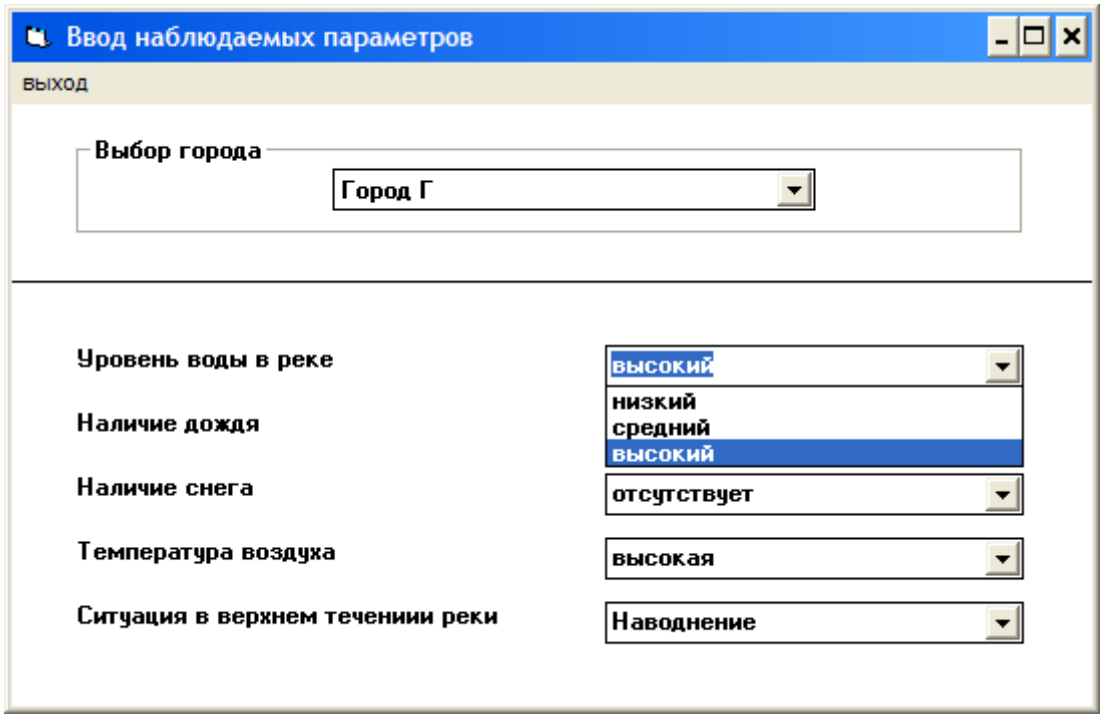


Рис. 26. Форма для ввода наблюдаемых параметров

Раздел 7. Экспертная система планирования последовательности операций робота с использованием предикатов

Постановка задачи

Найти последовательность операций робота, изменяющих исходное состояние мира в соответствии с целевым условием

Исходные данные

1. Описание исходного состояния мира представлено списком двухместных предикатов
2. Целевое условие задается также в виде списка двухместных предикатов
3. Имеется перечень базовых операций робота, которые производят модификацию предикатов

Предикат – синтаксическая конструкция, которая задает отношение между двумя компонентами

имя (операнд1, операнд2)

Здесь имя – идентификатор отношения, операнд – идентифицирует конкретный объект или переменную. В общем случае операнд указывать на некоторый объект из некоторого класса объектов

имя_класса(идентификатор_объекта)

Если класс состоит из одного объекта, то идентификатор можно не указывать, поскольку в этом случае объект можно идентифицировать уникальным именем класса.

Оператор – синтаксическая конструкция, которая описывает операцию, меняющую отношения между ее операндами

имя (операнд1, операнд2[,оперант3…])

Здесь имя – идентификатор операции; операнды – как правило, переменные, которым сопоставляются конкретные объекты.

Описание каждой операции содержит три списка предикатов: список предварительных условий, при наличии которых данная операция может быть выполнена; список предикатов, которые удаляются из описания мира после выполнения оператора; список предикатов, которые добавляются в описание мира после выполнения оператора.

Примеры использования введенных понятий

Предикаты:

1. Объект X находится на полу в помещении Y – at (X,Y)
2. Объект X расположен на объекте Y – on (X,Y) -> X находится в том же помещении, что и объект Y

Операторы:

Предикаты могут иметь префикс **not** – отрицание.
Символ перед именем «**?**» указывает на то, что это имя переменной.
Предикаты в списке предварительных условий расположены в порядке понижения приоритета.

1. Робот перемещается из помещения ?X в помещение ?Y – **move (?X,?Y)**
а. Список предварительных условий – at (робот,?X)
б. Список удалений – at (робот,?X)
в. Список добавлений – at (робот,?Y)

2. Робот перемещает одиночный объект ?X из помещения ?Y в помещение ?Z – **push (?X,?Y,?Z)**
а. Список предварительных условий –not (on (?A,?X)), not (on (?X,?A1)), at (?X,?Y), at (робот,?Y); где ?A, ?A1 – некоторые объекты (?X не находится на другом объекте ?A1 и на ?X не размещается другой объект ?A)
б. Список удалений – at (?X,?Y), at (робот,?Y)
в. Список добавлений – at (?X,?Z), at (робот,?Z)

3. Робот размещает одиночный объект ?X на объекте ?Y – **up (?X,?Y)**
а. Список предварительных условий – not (on (?A,?X)), not (on (?A1,?Y)), at (?X,?Z), at (робот,?Z); где ?Z – помещение, в котором находится объект ?Y, а ?A, ?A1 – некоторые объекты. В свою очередь, объект ?Y может располагаться на другом объекте ?A2 –
on (?Y,?A2) и тогда ?Z – помещение, в котором находится объект ?A2 и т.д.

б. Список удалений – at (?X,?Z)
в. Список добавлений – on (?X,?Y)

4. Робот снимает одиночный объект ?X с объекта ?Y – **down (?X,?Y)**
а. Список предварительных условий – not (on (?A,?X)), on (?X,?Y), at (робот,?Z); где ?Z – помещение, в котором находится объект ?Y, а ?A – некоторый объект. В свою очередь, объект ?Y может располагаться на другом объекте ?A1 - on (?Y,?A1)
б. Список удалений – on (?X,?Y)
в. Список добавлений – at (?X,?Z)

Примеры исходного состояния мира:

1. at (ящик1, склад), on (ящик2, ящик1), on (ящик3, ящик2), at (робот, склад)
2. at (ящик(1), склад), on (ящик(2), ящик(1)), at (ящик(3), склад), at (робот, склад), здесь исходное состояние мира записано с использованием класса объектов «ящик».

Несколько целевых условий, в которых целевые предикаты расположены в порядке понижения их приоритета:

1. on (ящик3, ящик1), at (ящик2, площадка), at (робот, склад)
2. at (ящик(?1), площадка), on (ящик(?2), ящик(?1)), at (робот, склад)
Здесь «?1» и «?2» указывают на любой идентификатор существующих объектов класса «ящик». Таким образом, с помощью переменных идентификаторов вида “?n” реализуется квантор общности – «для любого».

На рисунках 27, 28 показаны исходные и требуемые положения объектов и робота для этих целевых условий (на рис. 27 значения переменных идентификаторов ?1, ?2, ?3 выбираются из диапазона 1-3).

Описание состояния мира и целевое условие должны быть **полными** и не **противоречивыми**:
1. Для всех объектов и робота должно быть определено местоположение, т.е. в списке описания мира обязательно должны присутствовать предикат **at(робот,-)**, а все объекты **операнд2** в любом предикате **on(операнд1,операнд2)** - стоять на месте первого операнда хотя бы в одном из оставшихся предикатов.
2. Местоположение любого объекта должно быть определено только один раз: он может располагаться либо на полу в помещении, либо на каком-либо объекте.
3. Переменным в предикате или операторе нельзя присваивать одинаковые значения.
4. Для любого объекта в целевых условиях должен существовать объект в описании мира, с которым он совпадает или с которым его можно сопоставить (при наличии квантора «любой»). Т.е. целевое условие должны **соответствовать** описанию мира.

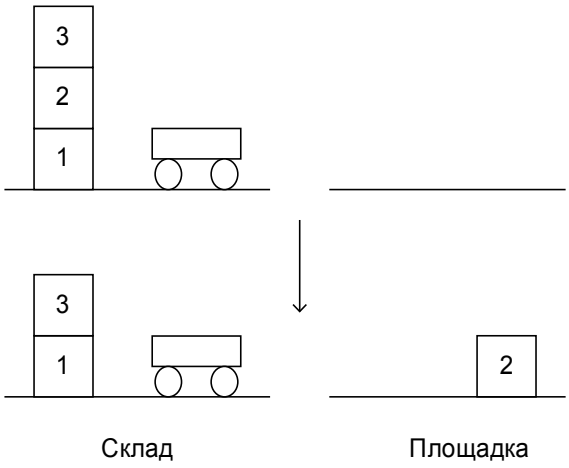


Рис. 27. Графическое представление начального и конечного состояния мира для примера 1

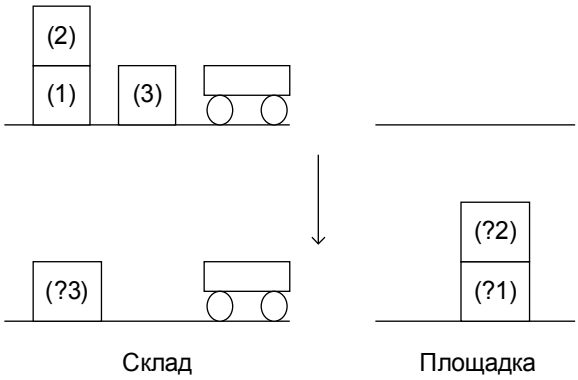


Рис. 28. Графическое представление начального и конечного состояния мира для примера 2

Алгоритм достижения поставленной цели – стратегия обратных рассуждений: от целей к подцелям. Алгоритм формирует список операторов, которые необходимо выполнить, для достижения поставленной цели.

1. Цикл по предикатам исходного целевого условия
   2. Очередной целевой предикат помещаем в стек
   3. Делай, пока стек не пуст
      3.1. Выбираем из стека целевой предикат
      3.2. Если у целевого предиката первый операнд имеет индекс вида «?n», то находим для этого операнда конкретный объект заданного класса из списка объектов, представленных в предикатах исходного описания мира
      3.3. Если в текущем описании мира такой предикат уже присутствует, то удаляем целевой предикат из стека
      3.4. Если в текущем описании мира подобный предикат отсутствует, то
         3.4.1. Ищем оператор, у которого в списке добавлений (или в списке удалений, если целевой предикат имеет префикс «not») имеется подобный предикат
         3.4.2. Присваиваем его переменным конкретные значения в соответствии с целевым предикатом

3.4.3. Остальным переменным, присутствующим в его предварительных условиях, присваиваем конкретные значения в соответствии с предикатами, присутствующими в текущем описании мира
3.4.4. FL_оператор=0 – все предварительные условия оператора выполнены
3.4.5. Цикл по списку предварительных условий
Если очередное предварительное условие не выполнено, то помещаем его в стек в качестве промежуточного целевого условия, присваиваем FL_оператор=1 и выходим из этого цикла
3.4.5. Конец цикла
3.4.6. Если FL_оператор=0, то
3.3.6.1. Помещаем оператор с конкретными значениями его переменных в список плана требуемой последовательности операторов
3.3.6.2. Корректируем текущую картину мира: удаляем предикаты, указанные в его списке удалений и добавляем предикаты из списка добавлений
3.4.6. Конец если
3.4. Конец если
3. Конец пока
1. Конец цикла

Замечания

1. Если целевой предикат содержит префикс «not», то в операторе соответствующий предикат необходимо искать в списке удалений.
2. Если операнд исходного целевого предиката содержит указание на любой элемент некоторого класса, то при поиске значения его переменного индекса выбирается тот экземпляр этого класс, который обеспечивает выполнение целевого предиката за минимальное число операций. При этом целевое условие следует формировать таким образом, чтобы к моменту обработки очередного целевого предиката его второй операнд был определен изначально или в результате обработки предыдущих целевых предикатов. Последнее имеет место для предиката **«on»**, если его второй операнд также снабжен индексом типа «любой» (см. выше целевое условие п.2).

Детализируем некоторые пункты предыдущего алгоритма.
Правила выбора оператора для выполнения целевого предиката:

1. Если целевой предикат **at (робот, ---)**, то оператор **move(?X,?Y)**
2. Если целевой предикат **at (---, ---)**, то оператор **push(?X,?Y,?Z)**
3. Если целевой предикат **on (---, ---)**, то оператор **up(?X,?Y)**
4. Если целевой предикат **not (on (---, ---))**, то оператор **down(?X,?Y)**

Правила определения значений переменным оператора и формирования подцелей, обусловленных не выполнением его предварительных условий:

1. Оператор **move(?X,?Y)**, целевой предикат **at (робот, B)** и предикат в описании мира **at (робот, B1)**.
Из сравнения целевого предиката и предиката из списка добавлений оператора - **?Y=B**
Согласно предикату в описании мира - **?X=B1**

Удаляем целевой предикат из стека, добавляем оператор в план и корректируем описание мира в соответствии с его списками удаляемых и добавляемых предикатов.

2. Оператор **push(?X,?Y,?Z)**, целевой предикат **at (C, B)**.
Из сравнения целевого предиката и предиката из списка добавлений оператора - **?X=C**, **?Z=B**
Если в описании мира присутствует предикат **on(D,C)**, то **?A=D** и подцель – **not(on(D,C))**
Если в описании мира присутствует предикат **on(C,D)**, то **?A1=D** и подцель – **not(on(C,D))**
Если в описании мира присутствует предикат **at(C,D)**, то **?Y=D**
Если в описании мира отсутствует предиката **at(робот,D)**, то подцель - **at(робот,D)**
Если все предварительные условия оператора выполнены, то удаляем целевой предикат из стека, добавляем оператор в план и корректируем описание мира в соответствии с его списками удаляемых и добавляемых предикатов.

3. Оператор **up(?X,?Y)**, целевой предикат **on (C, B)**.
Из сравнения целевого предиката и предиката из списка добавлений оператора - **?X=C**, **?Y=B**
Находим значение переменной **?Z**:

Fl=0
Buf=B
Пока **Fl=0**, делай:
Цикл по предикатам текущего состояния мира
Если текущий предикат **at(Buf,E)**, то **?Z=E, Fl=1**
Иначе Если текущий предикат **on(Buf,E1)**, то **Buf=E1**
Конец цикла
Конец пока

Если в описании мира присутствует предикат **on(D,C)**, то **?A=D** и подцель – **not(on(D,C))**
Если в описании мира присутствует предикат **on(D,B)**, то **?A1=D** и подцель – **not(on(D,B))**
Если в описании мира отсутствует предикат **at(C,E)**, то подцель – **at(C,E)**
Если в описании мира отсутствует предикат **at(робот,E)**, то подцель – **at(робот,E)**
Если все предварительные условия оператора выполнены, то удаляем целевой предикат из стека, добавляем оператор в план и корректируем описание мира в соответствии с его списками удаляемых и добавляемых предикатов.

4. Оператор **down(?X,?Y)**, целевой предикат **not (on (C, B))**.
Из сравнения целевого предиката и предиката из списка удалений оператора - **?X=C, ?Y=B**
Находим значение переменной **?Z** (см. п. 3)
Если в описании мира присутствует предикат **on(D,C)**, то **?A=D** и подцель – **not(on(D,C))**
Если в описании мира отсутствует предикат **at(робот,E)**, то подцель – **at(робот,E)**
Если все предварительные условия оператора выполнены, то удаляем целевой предикат из стека, добавляем оператор в план и корректируем описание мира в соответствии с его списками удаляемых и добавляемых предикатов.

Приведем развернутую последовательность действий, которые в соответствии с приведенным выше алгоритмом реализует первое целевое условие из приведенных выше.

1. Выбираем первый предикат из целевого условия «on (ящик3, ящик1)» и помещаем его в стек
2. Выбираем из стека последний помещенный предикат
3. Убеждаемся, что подобного предиката в описании мира нет
4. Находим оператор «up(?X,?Y)» , у которого аналогичный предикат «on (?X,?Y)» стоит в списке добавлений
5. Находим значения переменных, представленных в этом операторе:
- из сопоставления целевого предиката и добавляемого – «?X=ящик3», «?Y=ящик1»
- из сопоставления предикатов из списка предварительных условий оператора и описания мира – «?A1=ящик2», «?Z=склад»
6. Находим первое не выполненное предварительное условие – «not (on (ящик2,ящик1))» и помещаем его в качестве подцели в стек
7. Выбираем из стека последний помещенный предикат (имеет префикс «not»)
8. Находим оператор «down(?X,?Y)» , у которого аналогичный предикат «on (?X,?Y)» стоит в списке удалений
9. Сопоставляя соответствующие предикаты в целевом условии, в списке удалений, в списке предварительных условий и в текущем описании мира, находим значения переменных в этом операторе: «?X=ящик2», «?Y=ящик1», «?A=ящик3», «?Z=склад»
10. Находим первое не выполненное предварительное условие – «not (on (ящик3,ящик2))» и помещаем его в качестве подцели в стек
11. Выбираем из стека последний помещенный предикат (имеет префикс «not»)
12. Находим оператор «down(?X,?Y)» , у которого аналогичный предикат «on (?X,?Y)» стоит в списке удалений
13. Сопоставляя соответствующие предикаты, находим значения переменных в этом операторе: «?X=ящик3», «?Y=ящик2», «?Z=склад»
14. Поскольку все предварительные условия оператора **«down(ящик3,ящик2)»** выполнены, то:
- добавляем этот оператор в список плана требуемой последовательности операторов
- удаляем из описания мира предикат - «on (ящик3,ящик2)»
- добавляем в описание мира предикат – «at (ящик3,склад)»
- удаляем из стека обработанную подцель
15. Выбираем из стека подцель - «not (on (ящик2,ящик1))»
16. Находим оператор «down(?X,?Y)» , у которого аналогичный предикат «on (?X,?Y)» стоит в списке удалений
17. Сопоставляя соответствующие предикаты, находим значения переменных в этом операторе: «?X=ящик2», «?Y=ящик1», «?Z=склад»
18. Поскольку все предварительные условия оператора **«down(ящик2,ящик1)»** выполнены, то:
- добавляем этот оператор в список плана требуемой последовательности операторов
- удаляем из описания мира предикат - «on (ящик2,ящик1)»
- добавляем в описание мира предикат – «at (ящик2,склад)»
- удаляем из стека обработанную подцель
19. Выбираем из стека цель - «on (ящик3, ящик1)»
20. Находим оператор «up(?X,?Y)» , у которого аналогичный предикат «on (?X,?Y)» стоит в списке добавлений

21. Сопоставляя соответствующие предикаты, находим значения переменных в этом операторе: «?X=ящик3», «?Y=ящик1», «?Z=склад»
22. Поскольку все предварительные условия оператора **«up(ящик3,ящик1)»** выполнены, то:
- добавляем этот оператор в список плана требуемой последовательности операторов
- удаляем из описания мира предикат - «at (ящик3,склад)»
- добавляем в описание мира предикат – «on (ящик3,ящик1)»
- удаляем из стека обработанную цель
23. Поскольку стек пуст, берем следующий предикат из целевого условия «at (ящик2, площадка)» и помещаем его в стек
24. Выбираем из стека последний помещенный предикат
25. Убеждаемся, что подобного предиката в описании мира нет
26. Находим оператор «push(?X,?Y,?Z)» , у которого аналогичный предикат «at (?X,?Y)» стоит в списке добавлений
27. Находим значения переменных, представленных в этом операторе:
- из сопоставления целевого предиката и добавляемого – «?X=ящик2», «?Z=площадка»
- из сопоставления предикатов из списка предварительных условий оператора и описания мира – «?Y=склад»
28. Поскольку все предварительные условия оператора **«push(ящик2,склад,площадка)»** выполнены, то:
- добавляем этот оператор в список плана требуемой последовательности операторов
- удаляем из описания мира предикат - «at (ящик2,склад)»
- добавляем в описание мира предикаты – «at (ящик2,площадка)», «at(робот, площадка)»
- удаляем из стека обработанную цель
29. Поскольку стек пуст, берем следующий предикат из целевого условия «at (робот, склад)» и помещаем его в стек
30. Выбираем из стека последний помещенный предикат
31. Убеждаемся, что подобного предиката в описании мира нет
32. Находим оператор «move(?X,?Y)» , у которого аналогичный предикат «at (робот,?Y)» стоит в списке добавлений
33. Находим значения переменных, представленных в этом операторе:
- из сопоставления целевого предиката и добавляемого – «?Y=склад»
- из сопоставления предикатов из списка предварительных условий оператора и описания мира – «?X=площадка»
34. Поскольку все предварительные условия оператора **«move(площадка,склад)»** выполнены, то:
- добавляем этот оператор в список плана требуемой последовательности операторов
- удаляем из описания мира предикат - «at (робот,площадка)»
- добавляем в описание мира предикаты – «at (робот,склад)»
- удаляем из стека обработанную цель

Итак, список последовательности операций, приводящих к изменению мира в соответствии с первым целевым условием, выглядит следующим образом:
«down(ящик3,ящик2)», «down(ящик2,ящик1)», «up(ящик3,ящик1)», «push(ящик2,склад,площадка)», «move(площадка,склад)»

Аналогичным образом строится список последовательности операций и для второго целевого условия:
«push(ящик(3),склад,площадка)», «move (площадка,склад)»,

«down (ящик(2),ящик(1))», «push (ящик(1),склад,площадка)», «up(ящик(1),ящик(3))», «move (площадка,склад)»

Рассмотрим вопросы, связанные с программной реализацией экспертной системы планирования операций, выполняемых роботом при реализации целевого условия.

1. Общая структура для хранения конструкций предикатов

префикс	имя предиката	имя класса	ид. объекта	имя класса	ид. объекта	флаг

Здесь префикс =1 или 0, если префикс «not» перед предикатом отсутствует; “” – если идентификатор у операнда отсутствует; флаг=1, если предикат уже соответствует обработанному целевому условию, иначе флаг=0. Для предикатов, расположенных в списках предварительных условий, удалений или добавлений оператора, соответственно, имеют флаг=1,2,3. Знак «?» перед именем или идентификатором означает, что это наименование переменной. См. рис. 29, на котором приведена экранная форма для ввода и редактирования предикатов исходного состояния мира и целевого условия.

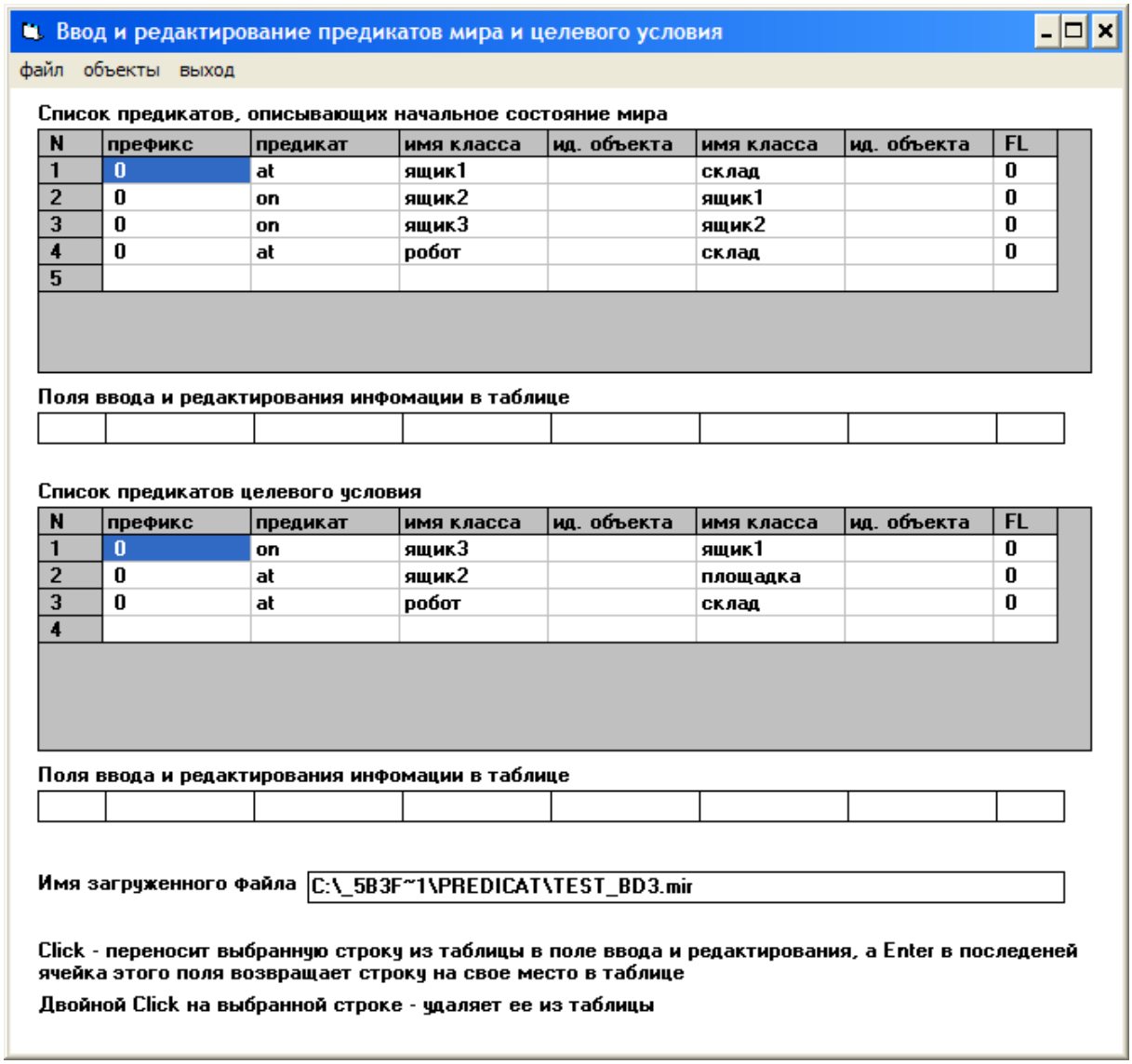


Рис. 29. Форма для ввода и редактирования предикатов мира и целевого условия

2. Структура для хранения заголовка оператора. По формату она совпадает со структурой п.1.

-1	имя оператора	арг 1	арг 2	арг 3		0

Здесь арг 1,2,3,4 – имена аргументов, “” – данный аргумент у оператора отсутствует.

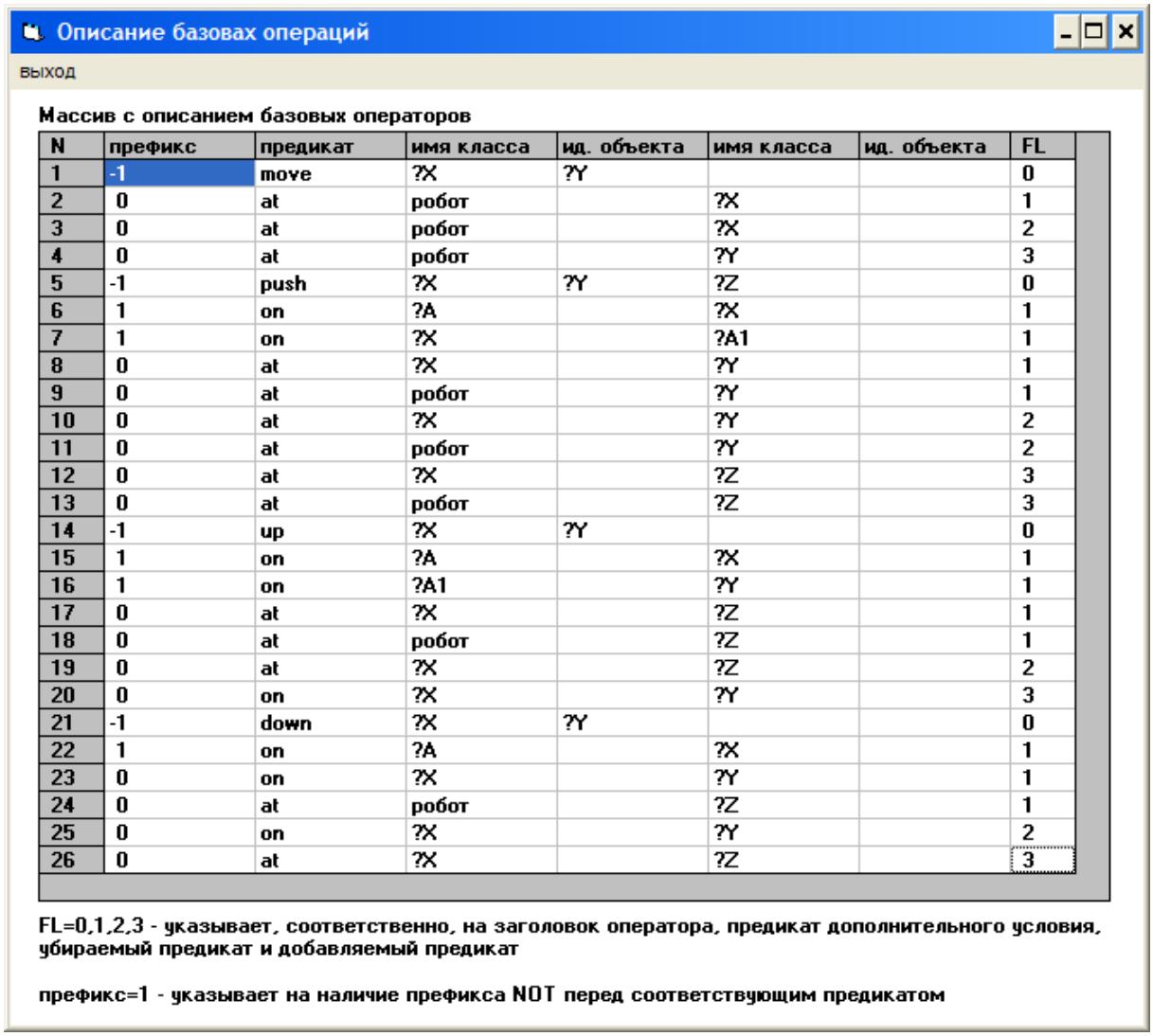
Используемые рабочие массивы указанных структур

1. peace – массив для хранения предикатов, описывающих текущее состояние мира
2. aim - массив для хранения предикатов, описывающих целевое условие
3. act – массив для хранения базовых операторов.

В нем каждый оператор представлен структурой заголовка оператора, за которой следует последовательность структур, описывающих предиката, включенные в его списки предварительных условий, удалений и добавлений (см. таблицу 1 и рис. 30, на котором представлена экранная форма программы для отображение базовых операторов со связанными с ними списками предикатов)

-1	имя оператора 1	арг 1	арг 2	арг 3	0	0
префикс	имя предиката	имя 1	ид 1	имя 2	ид 2	1
-	-	-	-	-	-	-
-	-	-	-	-	-	2
-	-	-	-	-	-	-
-	-	-	-	-	-	3
-	-	-	-	-	-	-
-1	имя оператора 2	арг 1	арг 2	арг 3	0	0
префикс	имя предиката	имя 1	ид 1	имя 2	ид 2	1
-	-	-	-	-	-	-
-	-	-	-	-	-	-

Таблица 1



N	префикс	предикат	имя класса	ид. объекта	имя класса	ид. объекта	FL
1	-1	move	?X	?Y			0
2	0	at	робот		?X		1
3	0	at	робот		?X		2
4	0	at	робот		?Y		3
5	-1	push	?X	?Y	?Z		0
6	1	on	?A		?X		1
7	1	on	?X		?A1		1
8	0	at	?X		?Y		1
9	0	at	робот		?Y		1
10	0	at	?X		?Y		2
11	0	at	робот		?Y		2
12	0	at	?X		?Z		3
13	0	at	робот		?Z		3
14	-1	up	?X	?Y			0
15	1	on	?A		?X		1
16	1	on	?A1		?Y		1
17	0	at	?X		?Z		1
18	0	at	робот		?Z		1
19	0	at	?X		?Z		2
20	0	on	?X		?Y		3
21	-1	down	?X	?Y			0
22	1	on	?A		?X		1
23	0	on	?X		?Y		1
24	0	at	робот		?Z		1
25	0	on	?X		?Y		2
26	0	at	?X		?Z		3

Рис. 30. Экран с таблицей базовых операций

4. plan – массив заголовков операторов сформированного плана имеет следующую структуру

имя оператора	имя класса	ид. объекта	имя класса	ид. объекта	имя класса	ид. объекта

На рис. 31, 32 приведена главная экранная форма программы с выведенными планами операций для двух рассмотренных выше примеров. На данной форме имеется переключать, позволяющий включить режим трассировки последовательности обрабатываемых целевых предикатов. Указанная трассировка отображается на экранной форме, представленной на рис. 33. Здесь в стеке собраны как исходные цели (с N=1), так и порождаемые ими подцели (с N>1).

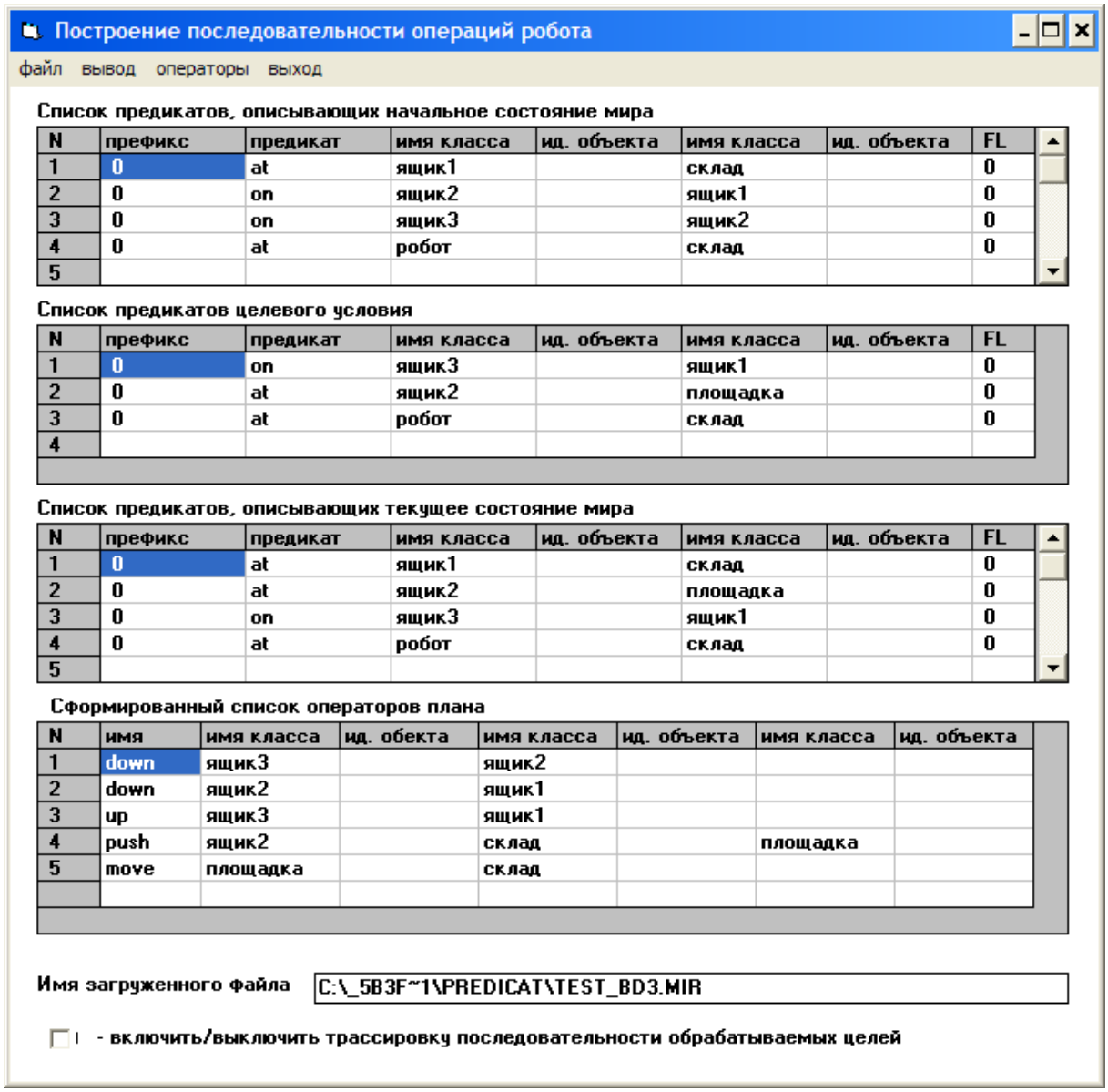


Рис. 31. Главная форма ЭС с построенным планом последовательности операций робота

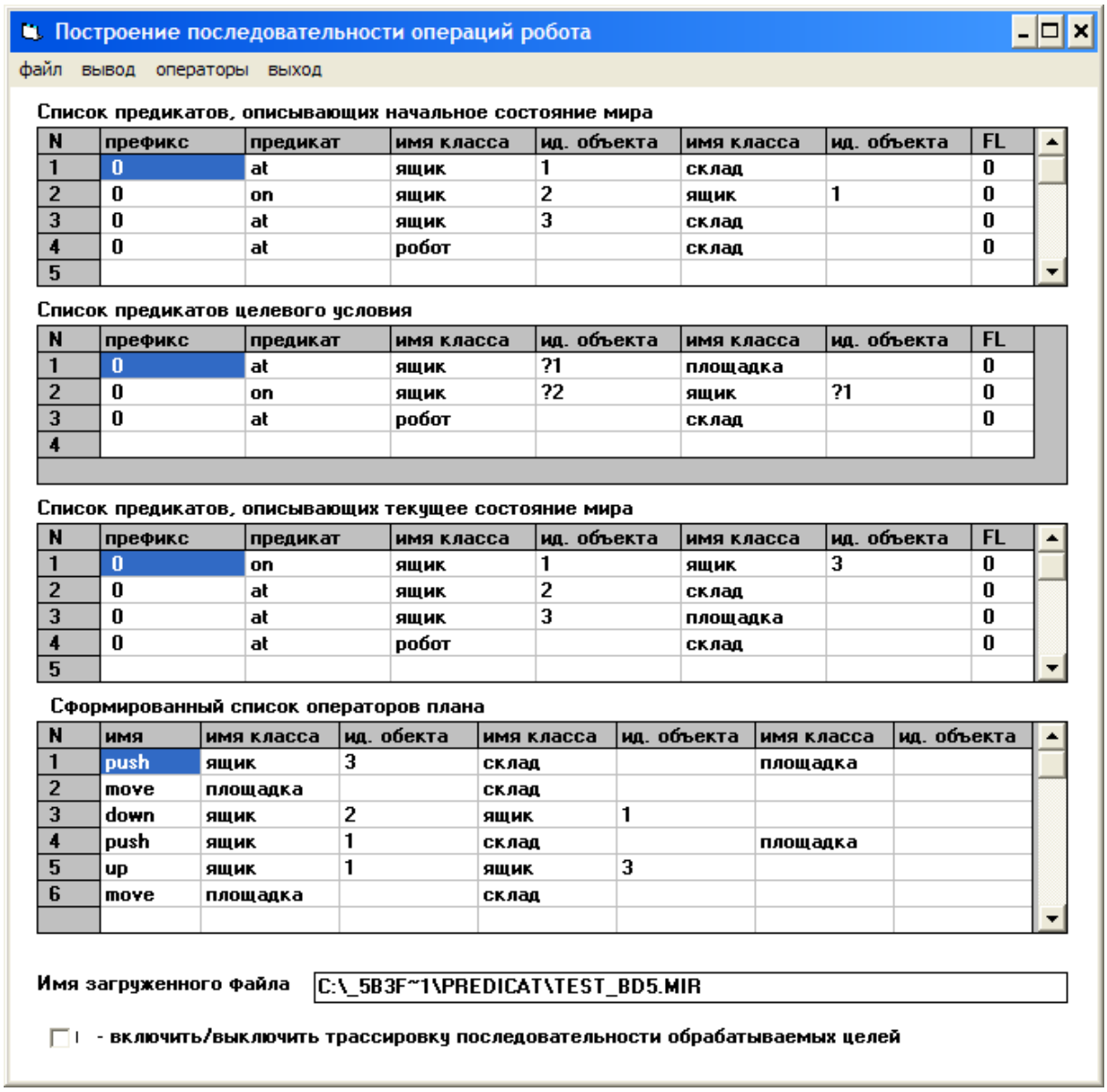


Рис.32. Второй пример работы ЭС планирования

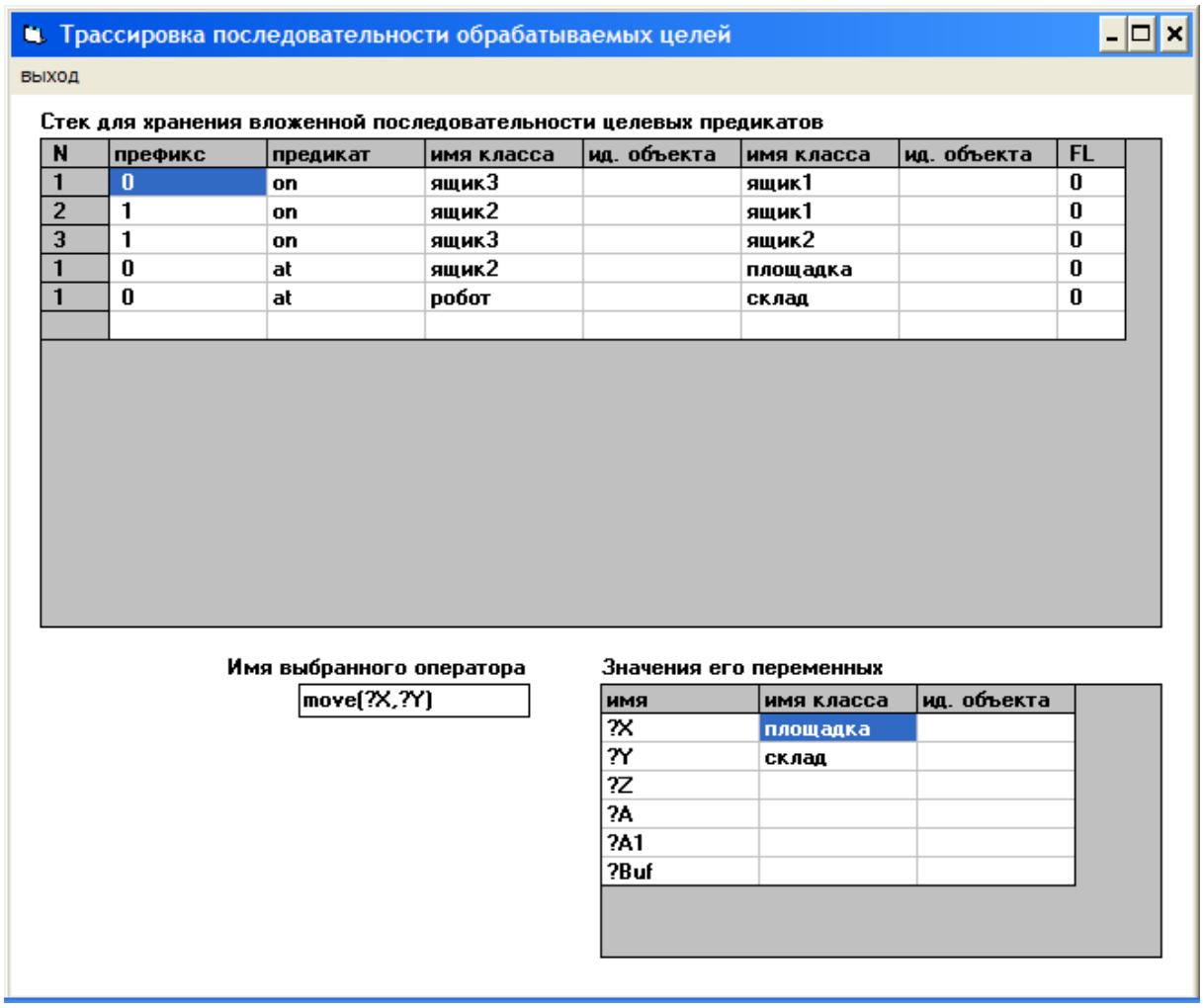


Рис. 33. Форма для просмотра трассировки процесса построения плана

Раздел 8. Экспертная система тестирования

Рассматривается ЭС, в которой БЗ состоит гипотез с приписанными диапазонами баллов и вопросов, снабженных альтернативными ответами (до 10 штук) с назначенными баллами. Пользователь последовательно отвечает на вопросы, каждый раз выбирая ответ из предложенного списка альтернатив. В конце тестирования ему предъявляется та гипотеза-заключение, в диапазон которой попадает суммарное число набранных баллов.

Данную ЭС можно использовать не только для прогонки разнообразных тестов, но и для оценки знаний в некоторой предметной области. В этом случае правильному ответу на вопрос следует назначать балл, соответствующий значимости данного вопроса или просто 1, а остальным альтернативным ответам – 0.

Главная форма приложения представлена на рис. 34.

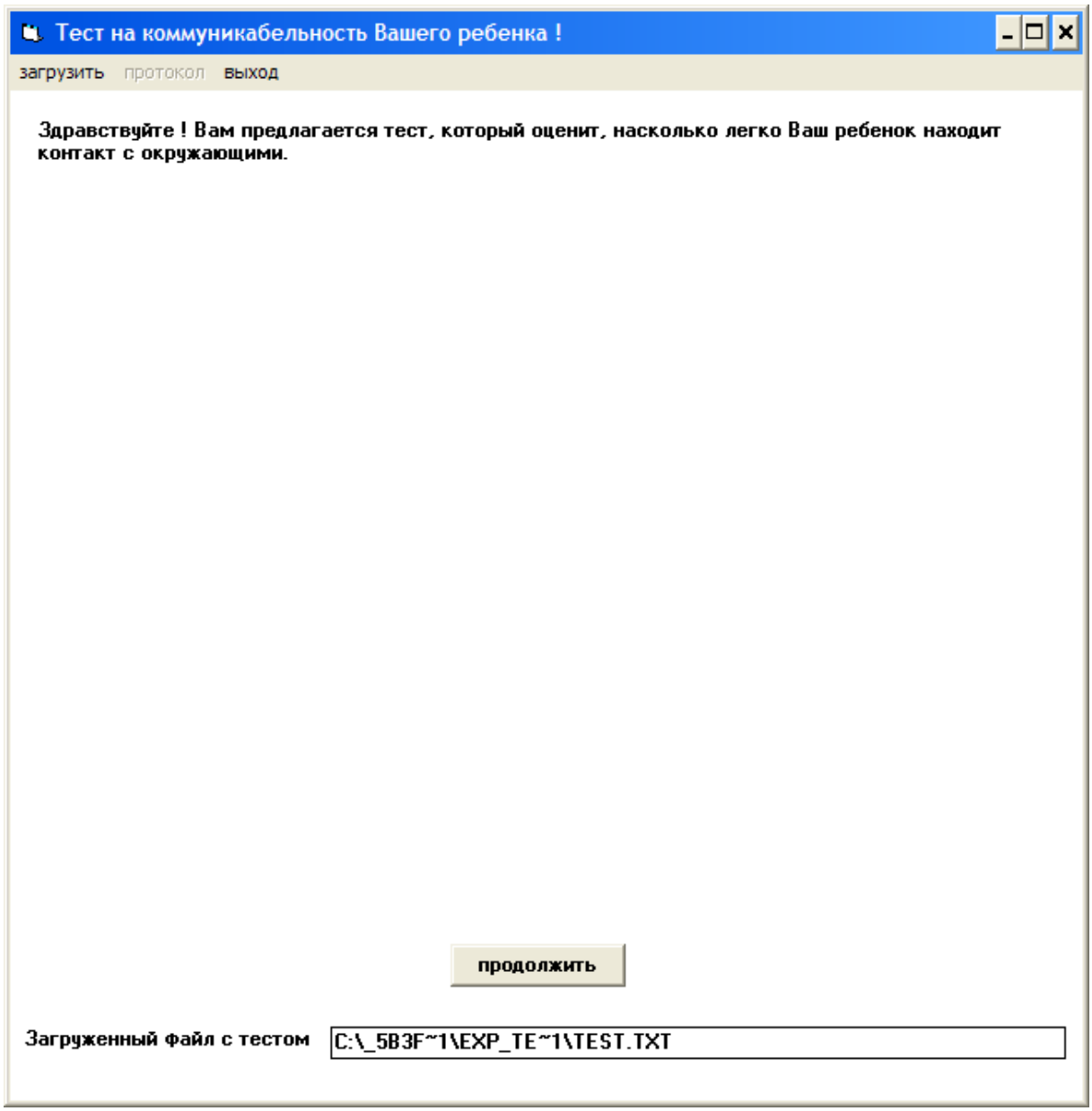


Рис. 34. Внешний вид главной формы программы.

В рассматриваемой программе тест загружается из файла формата .txt. Пример такого файла представлен на рисунке 35. Первая строка файла соответствует заголовку формы, вторая приветствию, которое появляется при загрузке файла с БЗ. Последующие строки файла задают общее число гипотез и сами гипотезы. Каждая гипотеза представлена двумя строками, определяющими верхнюю границу диапазона баллов, которые необходимо набрать пользователю для выбора данной гипотезы, и текст гипотезы. Нижняя граница диапазона начинается сразу за верхней границей последующей гипотезы. Сами гипотезы располагаются в порядке понижения их верней границы диапазона. Для последней гипотезы в списке нижняя граница диапазона равна 0.

В представленном на рисунке 35 примере, тест состоит из трех гипотез и 12 вопросов.

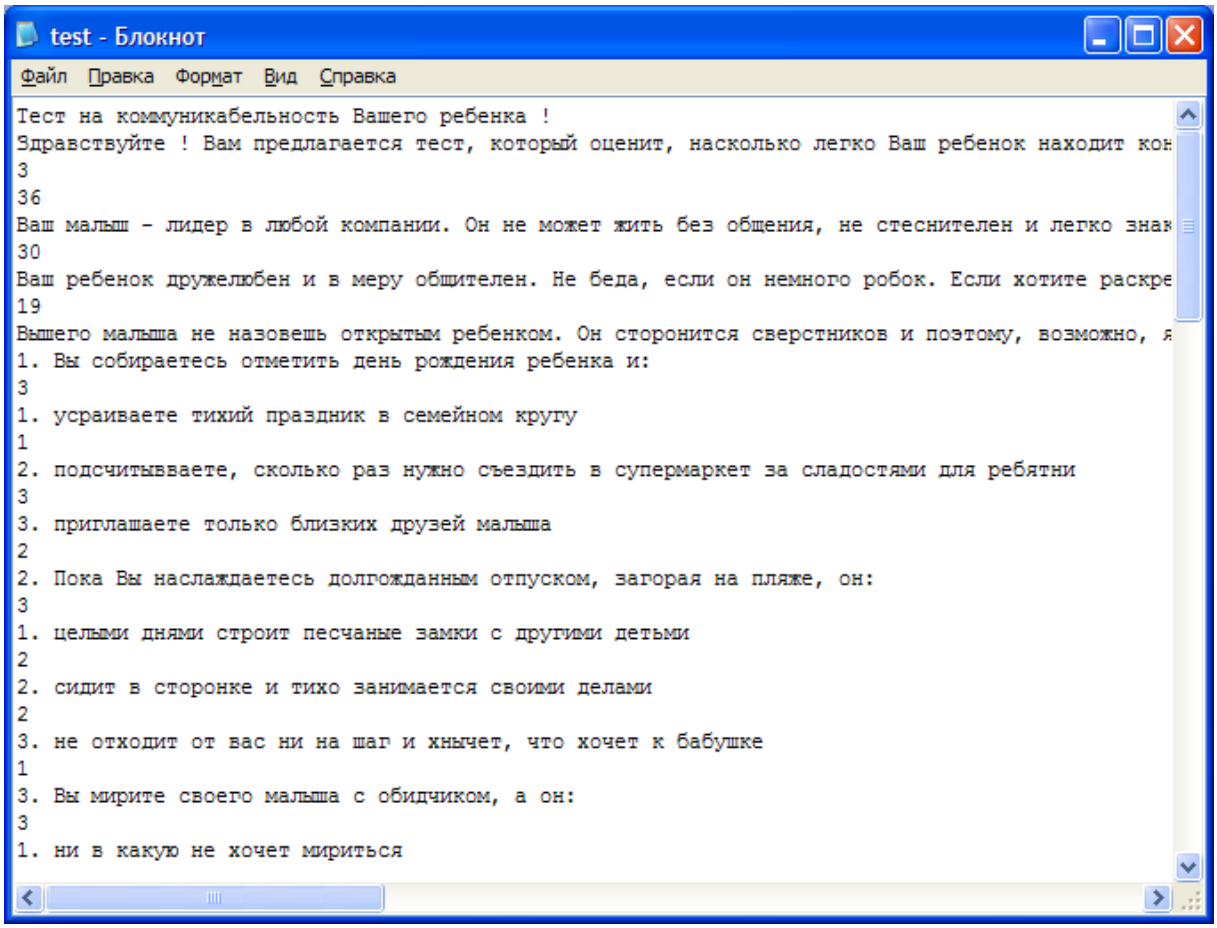


Рис. 35. Пример файла теста.

В рассматриваемом примере реализованы следующие интервалы: (0; 19) – «Вашего ребенка не назовешь открытым ребенком…», (20; 30) – «Ваш ребенок дружелюбен и в меру общителен…», (31; 36) – «Ваш малыш - лидер в любой компании….».

Далее следуют вопросы, число которых не регламентировано. Сначала идет строка, содержащая сам вопрос, затем строка, состоящая из числа, равного количеству вариантов ответа. Затем следуют варианты ответов. Каждый вариант записывается двумя строчками: с текстом ответа и числом баллов, ему приписанных.

Таким образом, составление файла теста не является трудоемкой задачей, следует лишь соблюдать следующие правила:

- в файле теста не должно быть пустых строк, ни в начале, ни в конце;
- должна быть соблюдена последовательность представления данных;
- после цифр, характеризующих оценку или правильный ответ не должно быть пробелов.

Создавать подобные файлы удобно с использованием стандартного текстового редактора - Блокнот. При этом проверка правильности кодирования БЗ осуществляется самим разработчика.

Каждый вопрос появляется в окне последовательно, после выбора пользователем ответа на предшествующий вопрос и нажатия кнопки «продолжить». Внешний вид программы при выбранном варианте ответа изображен на рисунке 36.

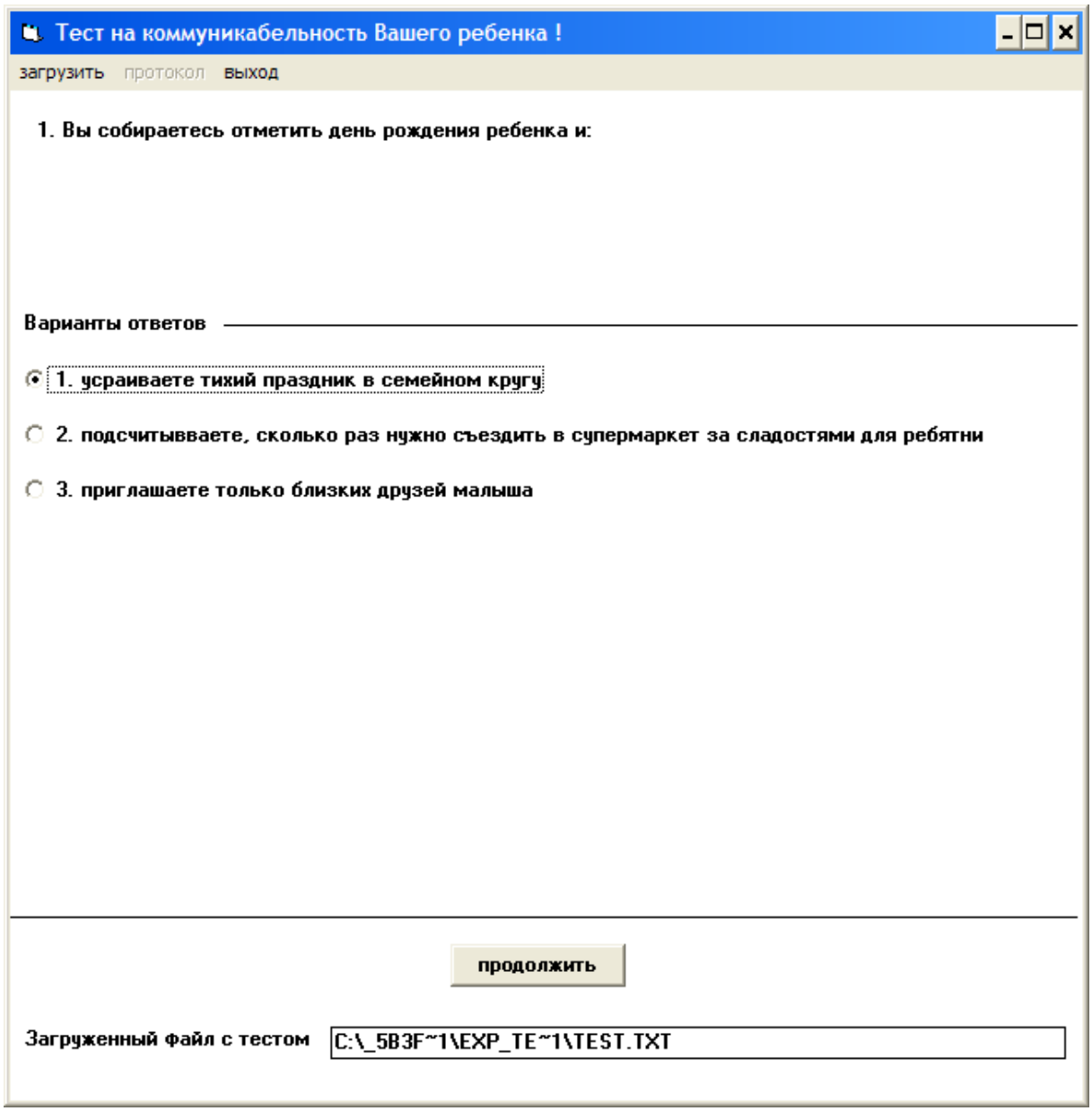


Рис. 36. Внешний вид программы при выбранном варианте ответа.

Пока пользователь не выбрал какой-либо ответ, кнопка «Продолжить» не отображается.

После прохождения всех вопросов, пользователю отображается результат тестирования (рис.37). Кроме выбранной гипотезы, указывается общее число вопросов, количество набранных баллов и время, затраченное на тестирование.

После выдачи результата становится доступным пункт меню «протокол». Он пользователю позволяет просмотреть протокол тестирование со списком вопросов и ответов (рис. 38). Таким образом, в ЭС реализована упрощенная подсистема «объяснения»

Опишем **алгоритм** работы программы.

В представленной программе текстовый файл обрабатывается последовательно. Вначале из файла считывается название теста, приветствие, число гипотез и гипотезы с их баллами. Название и приветствие сразу же выводятся на экран, а гипотезы и их баллы записываются в соответствующий буферный массив. Эти данные заносятся в

соответствующую структуру, состоящую из целой и строковой переменных. Одновременно отображается кнопка «продолжить».

Затем, после нажатия пользователем кнопки «продолжить», считывается вопрос, количество вариантов ответа, тексты вариантов и их баллы. Последние сохраняются в соответствующем массиве. Одновременно убирается кнопка «продолжить» до момента выбора ответа на поставленный вопрос. Вопрос и варианты ответов отображаются на экране с использованием управляющих элементов: метки и переключателей. Номер выбранного ответа сохраняется в отдельном массиве, который используется при построении протокола тестирования.

Последующие вопросы выводится аналогичным способом, пока не закончится загруженный файл. Далее происходит анализ результатов тестирования и выдается итоговая гипотеза. При этом убирается кнопка «продолжить» и делается доступным пункт меню «протокол».

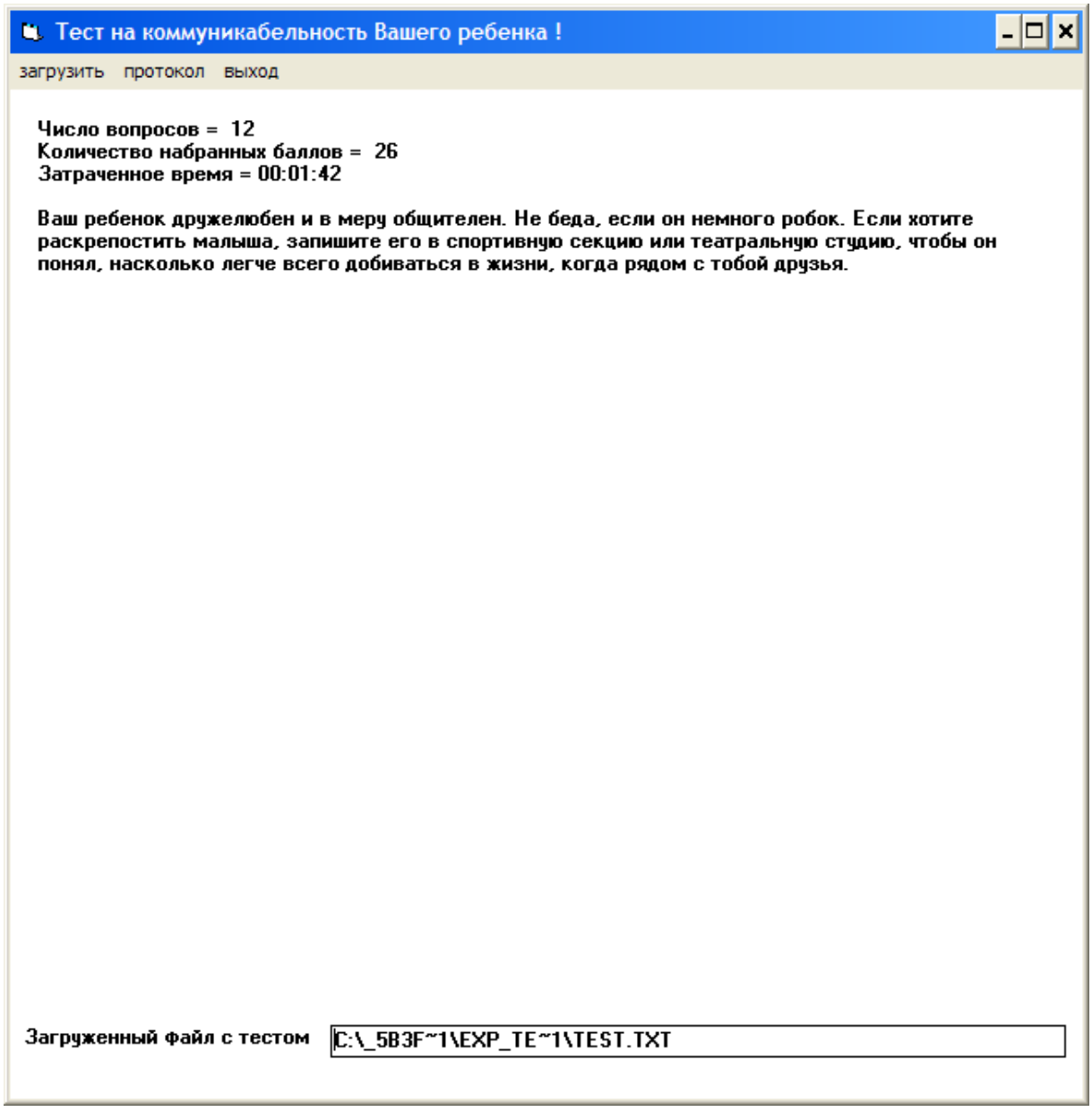


Рис. 37. Пример результата тестирования.

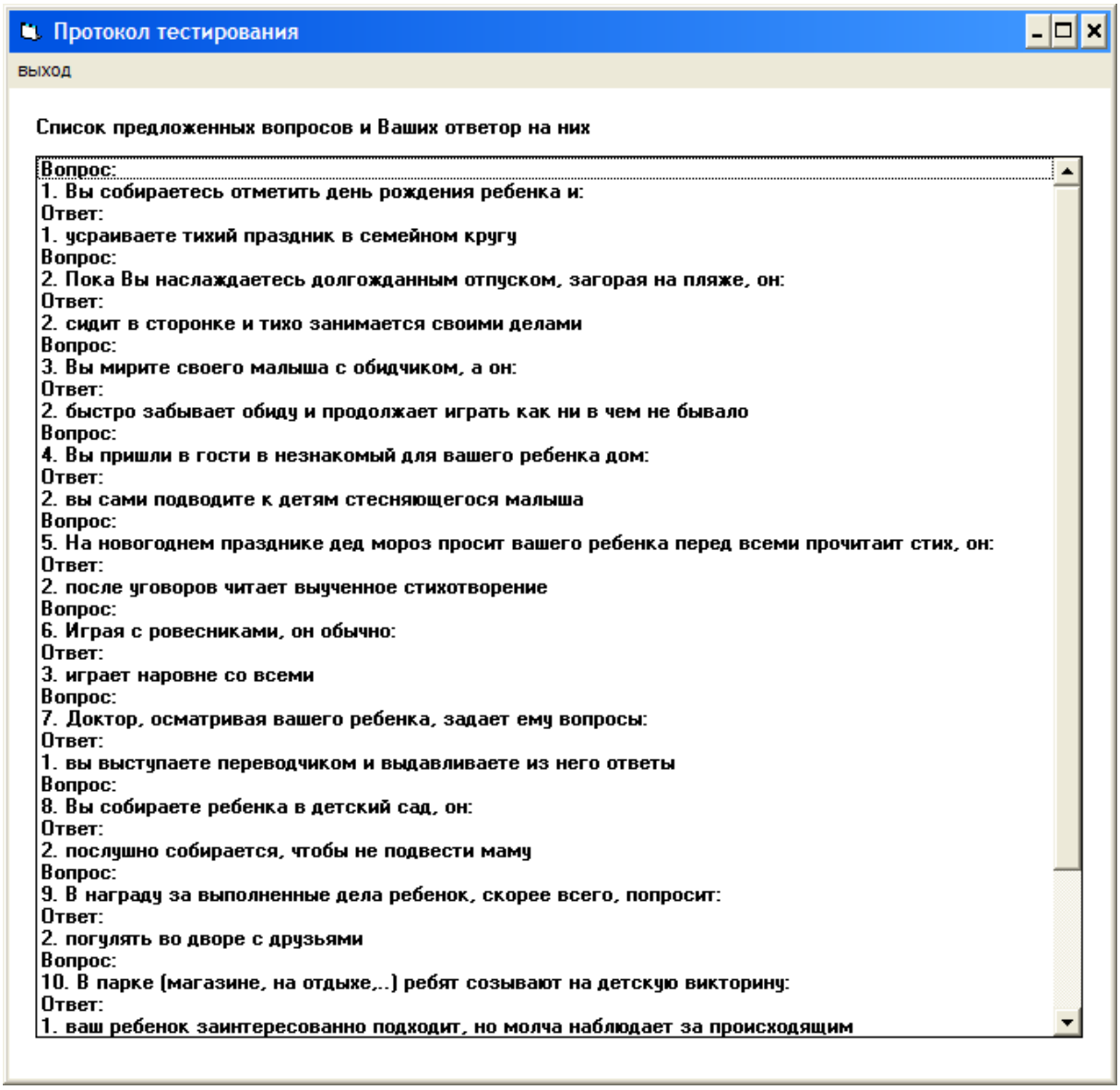


Рис. 38. Пример протокола тестирования.

Заполнение списка на форме рис. 38 производится следующим образом. Вновь открывается текстовый файл с БЗ, а затем из него выбираются записи с вопросами и соответствующими ответами, номера которых были сохранены во время тестирования.

По той же схеме построен другой вариант ЭС, в котором БЗ также состоит из гипотез и вопросов со списками альтернотивных ответов. В отличии от первого варианта в данном случае с каждым вариантом ответа связан список баллов, которые присваиваются гипотезам при выборе его пользователем. «Побеждает» та гипотеза, которая к концу тестирование имеет наибольший «коэффициент уверенности» (КУ) - отношение суммы набранных балов к максимально возможной сумме балов для данной гипотезы. Последняя сумма подсчитываются во время тестирования автомотически. Чтобы БЗ была **состоятельной,** все гипотезы должны иметь одно и то же максимально вожможное значение КУ.

Пример такой БЗ приведен в разделе 2 (пример 3). Формат ее кодировки в виде текстового файла, который используется в данном варианте ЭС, представлен на рис. 39.

Здесь также вначале расположены записи с заголовком для главной формы ЭС и приветствие. Затем — число гипотез и тексты сами гипотез. Их число не лимитировано. Далее следуют вопросы, каждый из которых систоит из следующих записей: текст вопроса, число альтернотивных ответов и соотвествующее число пар записей: текст ответа и список баллов для всех гипотез.

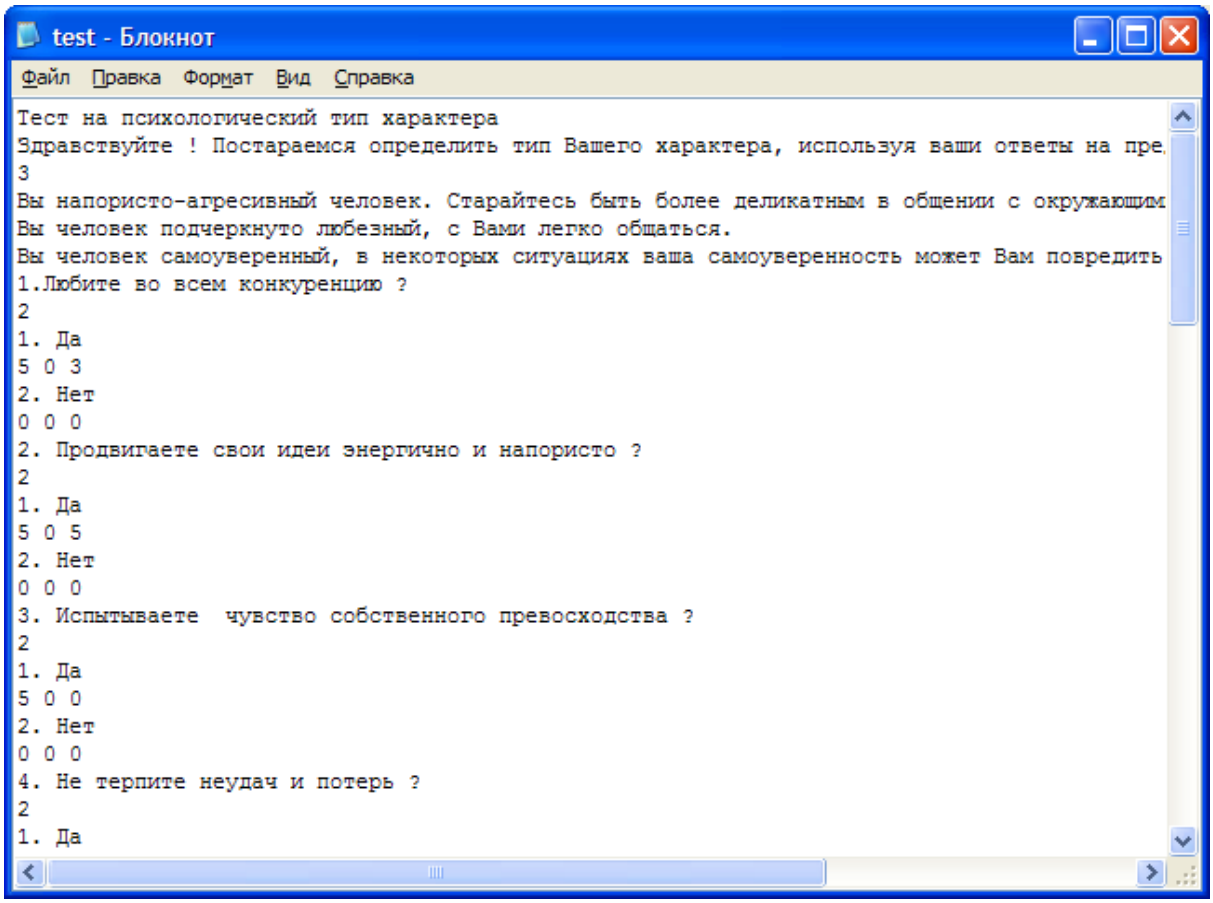


Рис. 39. Пример кодировки текстого файла с БЗ

Работа программы тестирования в целом следует прежнему алгоритму с небольшими изменениями:

- после выбора ответа и нажатия кнопки «продолжить» происходит корректировка набранных и максимально возможных баллов у всех гипотез
- по окончанию тестирования выводится следующая информация: число заданных вопросов, максимально возможное число балов у предъявляемой гипотезы, число набранных баллов, коэффициент уверенности, время тестирования и текст самой гипотезы (см. рис. 40).

Кроме того, по окончанию тестирования на форме становятся доступными пункты меню «список» и «протокол». При выборе первого выдается форма с ранжированным списком всех гипотез с набранными ими коэффициентами уверенности (см. рис. 41).

При формировании протокола отбираются только те вопросы и ответы, которые корректировали сумму набранных балов у «победившей» гипотезы (см. рис. 42).

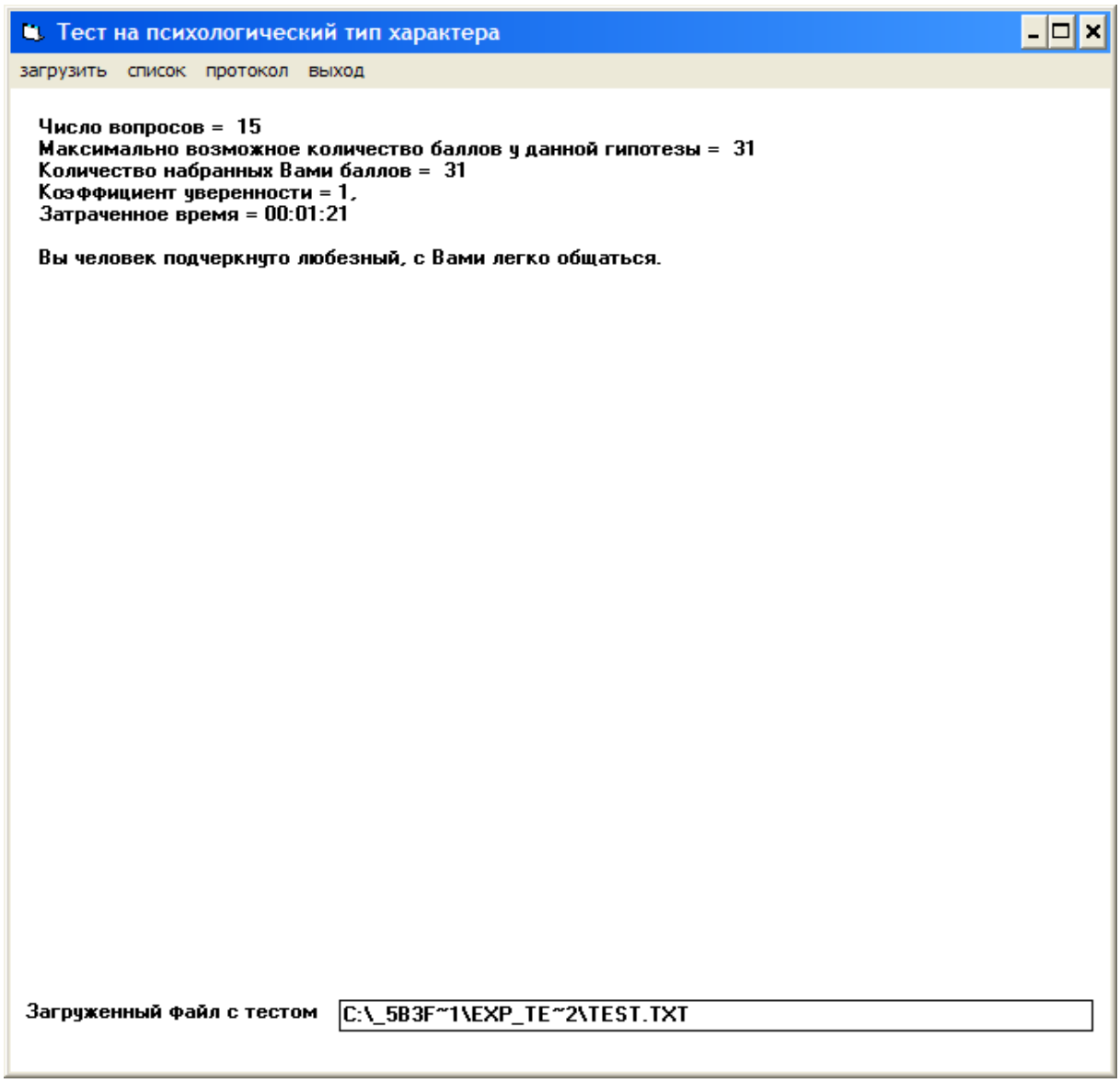


Рис. 40. Форма с результатами тестирования

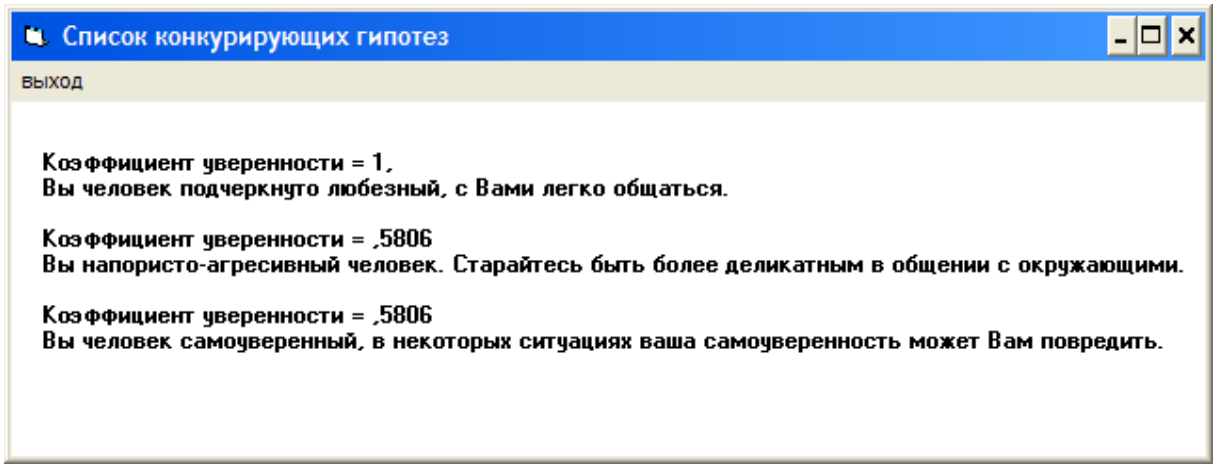


Рис. 41. Форма с окончательным ранжированным списком гипотез

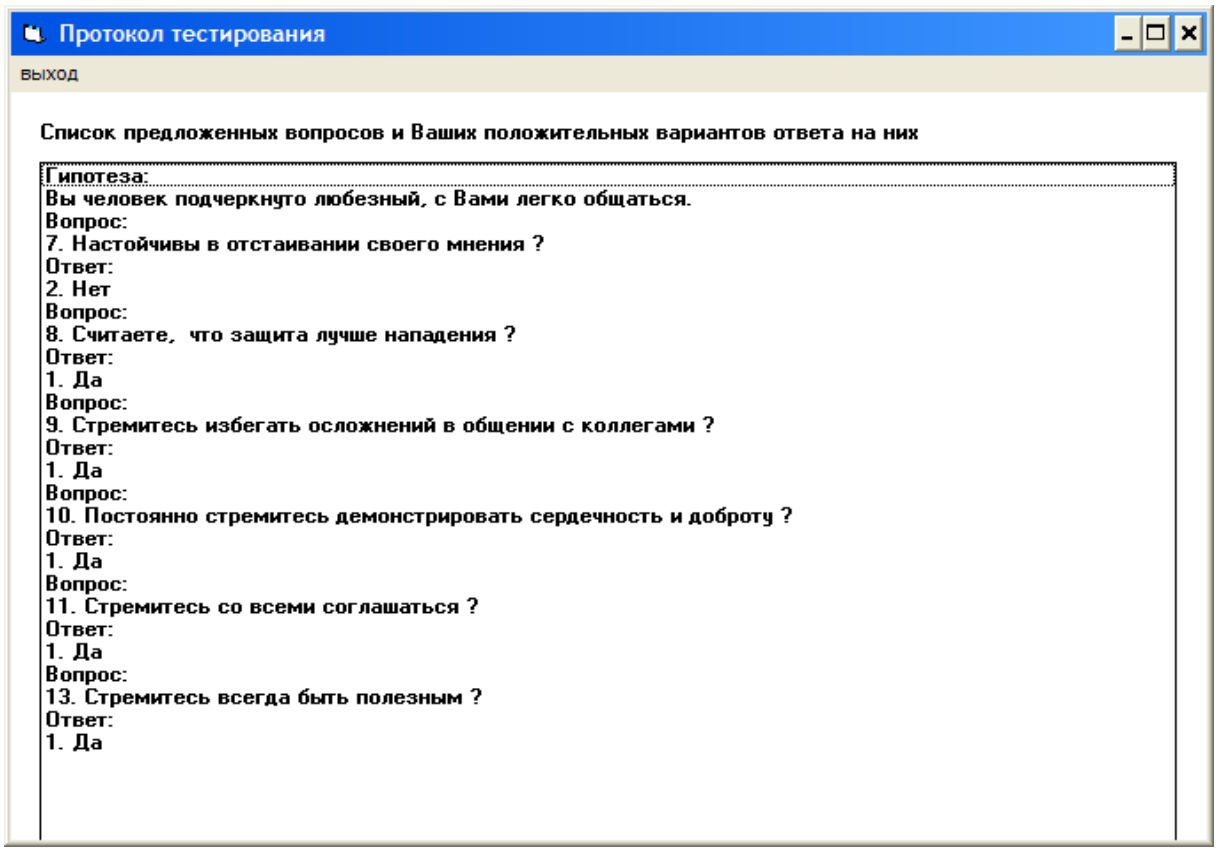


Рис. 42. Протокол со списком отобранных вопросов и выбранных ответов на них

Список литературы

1. Крис Нейлор. Как построить свою экспертную систему. М., Энергоатомиздат,1991
2. Фил Кокс. Как мы строим систему «Микроэксперт». В книге Экспертные системы. Принципы работы и примеры. Под ред. Форсайта Р., М., Радио и связь,1987, стр. 103-124
3. Левин Р., .Дранг Д., Эделсон Б. Практическое введение в технологию искусственного интеллекта и экспертных систем с иллюстрациями на бейсике. М.,Финансы и статистика, 1991
4. Питер Джексон Введение в экспертные системы. М. Издательский дом «Вильямс», 2001
5. Частиков А.П., Гаврилова Т.А., Белов Д.Л. Разработка экспертных систем. Среда CLIPS. СПб., БХВ-Петербург, 2003
6. Гаврилова Т.А., Червинская К.Р. Извлечение и структурирование знаний для экспертных систем. М., Радио и связь, 1992
7. Попов Э.В., Фоминых И.Б., Кисель Е.Б., Шапот М.Д. Статические и динамические экспертные системы. М., Финансы и статистика, 1996
8. Уотерман Д. Руководство по экспертным системам. М.: Мир, 1989
9. Построение экспертных систем. Под ред. Ф. Хейеса-Рота, Д. Уотермана, Д Лената. М.: Мир, 1987