Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Горно-Алтайский государственный университет»

РАБОЧАЯ ПРОГРАММА

дисциплины: **Интеллектуальные информационные системы**
Уровень основной образовательной программы: **бакалавриат**
Рекомендуется для направления подготовки 09.03.03 Прикладная информатика
профиль подготовки «Прикладная информатика в экономике»

Программа составлена в соответствии с требованиями ФГОС ВО
по направлению подготовки 09.03.03 Прикладная информатика
(утверждён приказом Минобрнауки от 12.03.2015 года № _207) и учебного плана по направлению подготовки 09.03.03 Прикладная информатика (профиль «Прикладная информатика в экономике»), утвержденного Ученым советом ГАГУ
(от 24.12.2015 протокол №12)

Рабочая программа утверждена на заседании кафедры экономики предприятия и прикладной информатики 26.01.16 г., протокол №5

1 Цель дисциплины

Изучение студентами проблематики и областей использования искусственного интеллекта в экономических информационных системах, освещение теоретических и организационно-методических вопросов построения и функционирования систем, основанных на знаниях, привитие навыков практических работ по проектированию баз знаний.

2 Место дисциплины в структуре ООП

Дисциплина «Интеллектуальные информационные системы» относится к дисциплинам по выбору профессионального цикла (Б3.В.ДВ.6.1).

Для освоения дисциплины «Интеллектуальные информационные системы» обучающиеся используют знания, умения, навыки, способы деятельности и установки, сформированные в ходе изучения предметов «Информатика и программирование», «Дискретная математика», «Информационные системы и технологии», «Пакеты прикладных программ», «Теория систем и системный анализ».

Основные положения дисциплины могут быть использованы при освоении дисциплин «Проектирование информационных систем», «Предметно-ориентированные экономические информационные системы», «Корпоративные информационные системы», «Аудит в компьютерной среде».

3 Требования к результатам освоения дисциплины

Процесс изучения дисциплины направлен на формирование следующих компетенций:

ПК-2	способностью разрабатывать, внедрять и адаптировать прикладное программное обеспечение

В результате изучения дисциплины студент должен:

знать назначение и классы ИИС; состав подсистем классов ИИС; модели и процессы жизненного цикла ИИС; стадии создания ИИС; технологии сбора, накопления, извлечения, структурирования, распространения и использования знаний; методы анализа прикладной области, решаемых задач, формирования требований к ИИС; методы и средства организации и управления проектом ИИС на всех стадиях жизненного цикла, оценка затрат проекта и экономической эффективности ИС; методы представления знаний; архитектуру СОЗ; методы и средства проектирования СОЗ, особенности создания БЗ;

уметь проводить анализ предметной области, выявлять информационные потребности и разрабатывать требования к ИИС; разрабатывать концептуальную модель прикладной области, выбирать инструментальные средства и технологии проектирования ИИС; проводить формализацию и реализацию БЗ.

владеть навыками работы с инструментальными средствами моделирования предметной области, прикладных процессов; использования функциональных и технологических стандартов ИИС; работы с инструментальными средствами проектирования БЗ.

4 Объем дисциплины и виды учебной работы

4.1 Распределение объема часов по видам учебной работы

Общая трудоемкость дисциплины составляет 3 зачетные единицы.

Вид учебной работы	Всего часов / зачетных единиц	Семестр
		6
Аудиторные занятия (всего), в том числе:	52/1,88	68
Лекции	14	20
Лабораторные работы (ЛР)	28	28
Практические занятия (ПЗ)	10	20
Самостоятельная работа (всего), в том числе:	56/1,55	40
Подготовка реферата	10	10
Конспектирование, подготовка к занятиям	30	20
Подготовка к промежуточному тестированию	16	10
Вид промежуточной аттестации (зачет, экзамен), объем, часы		Зачет с оценкой -
Общая трудоемкость, часы,	108	108
зачетные единицы	3	3

4.1 Матрица соответствия компетенций и разделов (тем) дисциплины

№ п/п	Наименование раздела (темы) дисциплины	Всего часов	Компетен -ции	Сумма компетенций
			ПК-2	
1	Информационные системы при интегрированном автоматизированном управлении экономическими объектами	5	+	1
2	Тенденции развития информационных систем поддержки принятия решений	5	+	1
3	Интеллектуальные технологии на основе инженерии знаний и искусственного интеллекта	6	+	1
4	Экспертные системы - системы, базирующиеся на знаниях	6	+	1
5	Представление знаний в интеллектуальных системах	6	+	1
6	Обработка знаний и вывод решений в интеллектуальных системах	6	+	1
7	Работа инженера знаний при разработке	5	+	1

	интеллектуальных систем			
8	Архитектура интеллектуальных информационных систем.	6		1
9	Инструментальные средства проектирования интеллектуальных систем	5	+	1
10	Интеллектуальная система планирования производства	5	+	1
11	Динамическая интеллектуальная система оперативно- диспетчерского управления предприятием.	6	+	1
12	Информационно-аналитические системы управления маркетингом.	5	+	1
13	Разработка интеллектуальной системы поддержки банковских решений	5	+	1
14	Экспертная система риск-менеджмента	5	+	1

5 Содержание дисциплины

5.1 Содержание разделов дисциплины

Тема 1. Информационные системы при интегрированном автоматизированном управлении экономическими объектами

Классификация информационных систем. Области применения информационных систем в экономике. Интегрированное автоматизированное производство, планирование и управление. Интегрированные экономические информационные системы.

Тема 2. Тенденции развития информационных систем поддержки принятия решений

Направления развития информационных технологий и систем. Моделирование и анализ ситуаций. Процесс подготовки и принятия решений. Новая технология решения задач управления в экономике. Интеллектуальные технологии.

Тема 3. Интеллектуальные технологии на основе инженерии знаний и искусственного интеллекта

Организация работы с данными и знаниями. Инженерия знаний. Развитие исследований в области искусственного интеллекта. Теория и практика искусственного интеллекта. Интеллектуальные информационные системы поддержки решений.

Тема 4. Экспертные системы - системы, базирующиеся на знаниях

Экспертные системы (ЭС) - основная разновидность ИИС. Функциональные возможности и характеристика ЭС. Области применения экспертных систем. Статические и динамические экспертные системы.

Тема 5. Представление знаний в интеллектуальных системах

Проблема представления и моделирования знаний; отличие знаний от данных; типы знаний; декларативные и процедурные модели представления знаний; основные модели представления знаний. Фреймовый подход, слоты, присоединенные процедуры. Семантические сети, отношения и объекты, вывод в семантической сети. Продукционные модели, компоненты продукционных систем. Логические модели представления знаний. Исчисление предикатов. Нечеткие множества. Представление и формализация нечетких знаний; нечеткие отношения. Нейронные сети. Нейрон как простой вычислительный элемент; персептрон; многослойные нейронные сети; ускоренное обучение в многослойных нейронных сетях; сеть Хопфилда; самоорганизующиеся нейронные сети.

Тема 6. Обработка знаний и вывод решений в интеллектуальных системах

Общие методы поиска решений в пространстве состояний в продукционных системах. Выводы на фреймах и в семантических сетях. Дедуктивные методы поиска решений. Поиск решений в условиях неопределенности. Вероятностная байесовская логика. Нечеткая логика и приближенные рассуждения. Вывод на нейронных сетях.

Тема 7. Работа инженера знаний при разработке интеллектуальных систем

Этапы проектирования интеллектуальных систем. Анализ специфики предметной области и методы приобретения знаний. Выявление, анализ и обработка источников знаний. Работа с экспертами. Проблема извлечения знаний. Автоматизация извлечения знаний и формирования модели.

Тема 8. Архитектура интеллектуальных информационных систем.

Структура системы. Проектирование базы знаний. Разработка механизма вывода решений. Объяснение и обоснование решений. Прогнозирование. Интеллектуальный интерфейс.

Тема 9. Инструментальные средства проектирования интеллектуальных систем.

Анализ и обзор традиционных языков программирования и представления знаний: язык Лисп; фрейм-ориентированный язык FRL; язык логического программирования Пролог; продукционный язык OPS.

Современные программные средства построения интеллектуальных систем. Объектно-ориентированный язык Visual Basic. Язык логического программирования Visual Prolog. Интегрированная инструментальная среда GURU. Интегрированная инструментальная среда G2 для создания интеллектуальных систем реального времени.

Тема 10. Интеллектуальная система планирования производства.

Производственная программа предприятия и календарное планирование. Задачи внутрифирменного планирования. Характеристика основных функциональных модулей. Использование имитационного моделирования в интеллектуальной системе для реализации расчетных функций и функций прогнозирования реализуемости планов. Учет неопределенности в системе.

Тема 11. Динамическая интеллектуальная система оперативно-диспетчерского управления предприятием.

Оперативно-диспетчерское управление предприятием. Специфика решаемых задач и реализуемых функций. Работа в реальном масштабе времени. Фреймово-продукционная модель представления знаний в системе. Имитационное моделирование на базе нечеткой сети Петри для реализации функций прогнозирования. Управление в реальном времени и оперативное прогнозирование.

Тема 12. Информационно-аналитические системы управления маркетингом.

Процесс управления маркетингом. Информационно-аналитические системы поддержки маркетинговых решений. CRM-технологии и интеллектуальный анализ данных в управлении маркетингом. Интернет-маркетинг и электронная коммерция.

Тема 13. Разработка интеллектуальной системы поддержки банковских решений

Анализ проблемной области. Архитектура системы и характеристика функциональных блоков. Основы банковского кредитования и методы определения кредитоспособности заемщика. Кредитный рейтинг. Моделирование бизнес-процессов в системе. Подсистема

оценки кредитоспособности заемщика. Анализ залоговых средств недвижимости и ценных бумаг. Формализация знаний и эвристик; построение базы знаний.

Тема 14. Экспертная система риск-менеджмента.

Цели, функции и структура системы. Информационная структура системы. Подсистема формирования портфеля ценных бумаг. Подсистема технического анализа рыночной информации и мониторинга портфеля. Разработка базы знаний фундаментального анализа. Система риск-менеджмента при управлении инвестиционным портфелем. Оценка рисков на основе фундаментального эвристического анализа. Интеллектуальный мониторинг рынка. Учет неопределенности и прогнозирование тенденций.

5.2 Распределение объема часов дисциплины по темам и видам занятий

№ п\п	Темы	Всего часов	Аудиторные занятия			Самост. работа
			лекции	практ. занятия	лабор. работы	
1	Информационные системы при интегрированном автоматизированном управлении экономическими объектами.	5	0	1	0	4
2	Тенденции развития информационных систем поддержки принятия решений.	5	0	1	0	4
3	Интеллектуальные технологии на основе инженерии знаний и искусственного интеллекта	6	1	1	0	4
4	Экспертные системы - системы, базирующиеся на знаниях	6	1	1	0	4
5	Представление знаний в интеллектуальных системах	6	1	1	0	4
6	Обработка знаний и вывод решений в интеллектуальных системах	6	1	1	0	4
7	Работа инженера знаний при разработке интеллектуальных систем	5	1		0	4
8	Архитектура интеллектуальных информационных систем	5	1		0	4
9	Инструментальные средства проектирования интеллектуальных систем	5		1	0	4
10	Интеллектуальная система планирования производства	6	2		0	4
11	Динамическая интеллектуальная система оперативно-диспетчерского управления предприятием.	6	1	1	0	4
12	Информационно-аналитические системы управления маркетингом.	5	1	1	0	3

13	Разработка интеллектуальной системы поддержки банковских решений	5	2		0	3
14	Экспертная система риск-менеджмента.	5	2		0	3
15	Основные понятия языка Пролог. Лабораторные работы «Программирование на языке Пролог»	32	0	1	28	3
	Всего по дисциплине	108	14	10	28	56
	Форма итогового контроля	**Зачет с оценкой**				

5.3 Практические занятия и лабораторный практикум

5.3.1 Практические занятия

Тема 1: Информационные системы при интегрированном автоматизированном управлении экономическими объектами.

Вопросы для контроля знаний на практическом занятии:

1.Классификация информационных систем.

2.Области применения информационных систем в экономике.

3.Интегрированное автоматизированное производство, планирование и управление.

4.Интегрированные экономические информационные системы.

Задание. Студенты в течении 15 минут пишут словарный диктант. В работу включены понятия, которым студенты должны дать определение (ИТ, ИС, АСУ, информационная система для департаментов, ИСМ, СППР, ИИС, оперативные информационные системы)

Тема 2: Тенденции развития информационных систем поддержки принятия решений.

Вопросы для контроля знаний на практическом занятии:

1. Направления развития информационных систем.

2. Моделирование и анализ ситуаций.

3. Процесс подготовки и принятия решений.

4. Новая технология решения задач управления.

Задание. Студенты в течение 15 минут пишут словарный диктант. В работу включены четыре понятия, которым студенты должны дать определение (мультимедиа и виртуальная реальность, модель, аналоговая модель, решение с несколькими альтернативами, математическое программирование, вероятностные имитационные модели, интеллектуальное поведение, интеллектуальная технология).

Доклады:

1.Инструментарий создания информационной системы с Web-интерфейсом.

2.Система проверки бухгалтерского баланса на достоверность в процессе мониторинга и анализа финансового состояния неплатежеспособных предприятий.

3.Интеллектуальный анализ данных при управлении маркетингом в торгово-производственной фирме.

4.Интеллектуальный анализ данных при мониторинге технологического процесса в информационной системе управления производством.

Тема 3: Интеллектуальные системы на основе инженерии знаний и искусственного интеллекта.

Вопросы для контроля знаний на практическом занятии:

1. Организация работы с данными и знаниями. Инженерия знаний.
2. Развитие исследований в области искусственного интеллекта.
3. Теория и практика искусственного интеллекта.
4. Интеллектуальные информационные системы поддержки решений.

Задания для обсуждения на основе самостоятельной подготовки студентов

1. Что в себя включают данные в информационной системе?
2. Назовите и охарактеризуйте три основных источника, получение данных для информационной системы?
3. Охарактеризуйте структуру данных и систем управления базами данных?
4. Охарактеризуйте хранилище данных?
5. OLAP: оперативная аналитическая обработка данных?
6. Интеллектуальный анализ данных?
7. Интеллектуальные базы данных?
8. Знания в искусственном интеллекте. База знаний?
9. Система управления базой знаний?
10. Обработка знаний?
11. Инженерия знаний?
12. Эволюция исследований в области искусственного интеллекта?
13. Схематическое представление интеллектуальной СППР?

Тема 4: Экспертные системы - системы, базирующиеся на знаниях

Вопросы для контроля знаний на практическом занятии:

1. Экспертные системы – основная разновидность интеллектуальных систем.
2. Функциональные возможности и характеристика ЭС.
3. Области применения экспертных систем.
4. Стратегические и динамические ЭС.

Задание 1. Студенты в течение 15 минут пишут словарный диктант. В работу включены понятия, которым студенты должны дать определение (экспертная система, экспертиза, эксперт, извлечение знаний, база знаний, механизм вывода, рабочая область, подсистема верификации и совершенствования знаний, системы предсказания).

Доклады:

1. Концепция автоматизированной интеллектуальной системы анализа фондового рынка.
2. Система анализа рисков при управлении инвестиционным портфелем и кредитами в информационной системе “Банк”.
3. Методы анализа финансовых инструментов и прогнозирования с использованием нейронных сетей.
5. Система управления инвестиционным портфелем на основе генерации и анализа гипотез с использованием Байесовского подхода.

Тема 5: Представление знаний в интеллектуальных системах

Вопросы для контроля знаний на практическом занятии:

1. Проблемы представления и моделирования знаний.
2. Представление знаний на основе фреймов и семантических сетей.
3. Продукционные и логические модели представления знаний.
4. Представление и формализация нечетких знаний.
5. Нейронные сети.

Задание 1. Студенты в течение 15 минут пишут словарный диктант. В работу включены четыре понятия, которым студенты должны дать определение (фреймы, семантические сети, продукционные модели, предметными константами).

Тема 6: Обработка знаний и вывод решений в интеллектуальных системах.

Контрольная работа

Вариант № 1

Понятие: Поиск в глубину, поиск в ширину, поиск с возвратом, хронологическим возвратом.

Вопросы:

1. Общие методы поиска решений в пространстве состояний?
2. Методы вывода на основе прямой и обратной цепочек?
3. Эвристические методы поиска?
4. Метод редукции?

Вариант № 2

Понятие: граф редукции задач, фреймовая система, имя слота, процедурные семантические сети.

Вопросы:

1. Методы поиска решений в больших пространствах состояний?
2. Процедуры – демоны и присоединенные процедуры (методы или служебные)?
3. Взаимодействие фреймов и правил.
4. Вывод в семантических сетях.

Тема 7: Инструментальные средства проектирования интеллектуальных систем.

Вопросы для контроля знаний на практическом занятии:

1. Анализ традиционных языков программирования и представления знаний.

1.1. Специализированный язык LISP.

1.2 Фрейм-ориентированный язык FRL.

1.3 Язык логического программирования PROLOG.

1.4. Продукционный язык OPS.

2. Современные программные средства построения интеллектуальных систем.

2.1. Объектно-ориентированный язык Visual Basic.

2.2. Язык логического программирования Visual Prolog.

2.3. Интегрированная инструментальная среда GURU.

2.4. Интегрированная инструментальная среда G2 для создания интеллектуальных систем реального времени.

Тема 8: Динамическая интеллектуальная система оперативно-диспетчерского управления предприятием.

Вопросы для контроля знаний на практическом занятии:

1. Оперативно-диспетчерское управление предприятием
2. Функционирование интеллектуальной системы диспетчерского управления в составе ИСУП в реальном времени.
3. Представление знаний и вывод решений в системе.
4. Имитационная прогнозирующая модель для оперативных интервалов времени

4.1 Принципы построения имитационной прогнозирующей модели.

4.2 Построение имитационной модели производства на базе Сети Петри.

4.3 Моделирование неопределенности ситуаций в имитационной прогнозирующей модели

4.4 Процедуры принятия решений при диспетчерском управлении во взаимодействии с имитационной прогнозирующей моделью ИИС.

Доклады:

1. Автоматизированная система документооборота банка.
2. Имитационная модель транзакционных процессов коммерческого банка.

3. Разработка интеллектуальной технологии Интернет-маркетинга.
4. Разработка модели реинжениринга бизнес-процессов в банке.
5. Разработка мультиагентных технологий в интеллектуальной информационной системе.

Тема 9: Информационно-аналитические системы поддержки решений в маркетинге.

Вопросы для контроля знаний на практическом занятии:

1. Процесс управления маркетингом.
2. Информационно – аналитические системы поддержки маркетинговых решений.
3. CRM – технологии и интеллектуальный анализ данных в управлении маркетингом.

4 Интернет – маркетинг и электронная коммерция.

Задание 1. Студенты в течение 15 минут пишут словарный диктант. В работу включены четыре понятия, которым студенты должны дать определение (стратегия управления маркетингом, МИС, механизм репликации, связь на базе документов).

Тема 10: Основные понятия языка Пролог.

Вопросы для изучения и контроля знаний на практическом занятии:

1. Структура программы на языке Пролог
2. Управление поисковым решением
3. Использование списков в Прологе
4. Внутренние базы данных
5. Создание приложения
6. Использование файлов в Прологе

Задание 1. Студенты в течение 15 минут пишут словарный диктант.
Далее решают логические задачи. Изучают понятия и определения языка Пролог, отвечают на вопросы по темам лабораторных работ.

5.3.2 Лабораторный практикум «Программирование на языке Пролог»

Лабораторная работа 1

Структура программы на языке Пролог

Программа на Прологе состоит из утверждений (clauses). Имеются два вида утверждений: факты и правила.

Факт – это утверждение которое заведомо истинно. Например:

```
likes(elena,tennis).      /*Лене нравится теннис*/
likes(tom,swimming). /*Тому нравится плавать*/
student(dima).            /*Дима – студент*/
```

Каждый факт представляет собой отношение, связывающее объекты. Символическое имя отношения называется именем предиката или **предикатом**. Объекты отношения называются аргументами. В приведенных примерах likes – имя предиката, elena и tennis – аргументы предиката.

Правило в Прологе используется для получения вывода на основе имеющихся в программе фактов. Например:

```
likes(bill,X) if likes(tom,X).  /* Биллу нравится нечто, что нравится Тому */
```

Правила в Прологе состоят из трех частей:

- заголовка;
- слова if или символов ":-";
- тела утверждения.

Заголовок – это факт, который должен быть истинным, если истинно некоторое число условий.

Слово if или последовательность символов ":-" отделяют заголовок от тела утверждения.

Тело утверждения – это множество условий, которые должны быть истинны, если требуется доказать истинность заголовка. Условия могут быть соединены словом and (и) или or (или). Вместо слова and часто используют запятую, а вместо слова or – точку с запятой. Заголовок правила также называют левой частью правила, а условия, следующие за символом ":-" – правой частью правила.

Переменные в Прологе используются для обозначения неопределенных объектов. Имя переменной должно начинаться с заглавной буквы латинского алфавита или символа подчеркивания "_", после которых могут следовать любое число букв, цифр и символов подчеркивания. Переменные с таким обозначением называются именованными. Если переменная обозначена одним символом подчеркивания "_", то она является анонимной. Анонимная переменная используется, когда ее значение не представляет интереса. Переменные в Прологе локальны. Например, если два утверждения содержат переменную X, то это две различные переменные.

Программа на Прологе состоит из разделов. Основные разделы:

```
domains    /* раздел описания доменов*/
predicates /*раздел описания предикатов */
goal       /* целевое утверждение */
clauses    /* раздел утверждений */
constants  /* раздел констант */
```

Другие разделы будут рассмотрены позднее. В программе могут отсутствовать некоторые разделы. С другой стороны программа может содержать несколько разделов constants, domains, predicates, или clauses. Константы, домены и предикаты должны быть определены до их использования. Комментарии в программе обрамляются символами /* и */. Комментарии можно помещать в любом месте программы.

Главной частью программы является **раздел утверждений**. В разделе clauses перечисляются все факты и правила, имеющиеся в программе. Каждое утверждение заканчивается точкой. Все утверждения, которые описывают один и тот же предикат, должны идти одно за другим.

В **разделе описания предикатов** (predicates) определяются имена предикатов и типы его аргументов. Если в программе используется встроенный предикат, то его описывать не нужно.

Имя предиката должно начинаться с буквы. Далее могут следовать буквы (латинские, прописные и строчные), цифры и знак подчеркивания в произвольной последовательности. Не рекомендуется использовать заглавные буквы в качестве первой буквы имени предиката (другие версии Пролога это не разрешают). Имя может иметь длину до 250 символов.

Типы аргументов перечисляются через запятую и обрамляются круглыми скобками. Типами аргументов предиката могут быть либо стандартные домены, либо домены, объявленные в секции domains. В отличие от предложений в секции clauses описание предиката не заканчивается точкой.

Например, возможно следующее описание предиката likes:

```
predicates
   likes(symbol,symbol)
```

Для одного предиката может быть несколько описаний. Например,

```
predicates
   likes(name,activity)
   likes(name,thing)
```

В данном случае домены name, activity, thing определяются пользователем.

Арностью предиката называется количество аргументов, которые он имеет. Можно объявить предикаты с одинаковыми именами, но различными арностями:

```
predicates
   student(name)
   student(name, university)
```

Когда для предиката имеется несколько описаний, они должны идти одно за другим.

Если используемые вами аргументы не относятся к стандартным типам данных, то их тип должен быть описан в **секции доменов**. Домены в Прологе сходны с описанием типов в Паскале. Вы можете давать свои собственные имена различным типам данных. Например:

```
domains
   name,activity=symbol    /* имя, род деятельности */
predicates
   likes(name,activity)
```

Такое объявление обычно используется для синтаксически похожих объектов, но различающихся семантически. Использование специальных доменов улучшает понимание программы и позволяет Прологу контролировать домены, но при этом аргументы с типами из специальных доменов не могут смешиваться. Не смотря на то, что name и activity – symbol, эти аргументы принадлежат различным типам данных.

В приведенной ниже таблице перечислены основные стандартные домены Пролога.

Таблица2

Домен	**Возможные значения**
Short	-32768 .. 32767
Ushort	0 .. 65535
long	-2147483648 .. 2147483647
ulong	0 .. 4294967295
integer	-2147483648 .. 2147483647
unsigned	0 .. 4294967295
byte	0 .. 255
word	0 .. 65535

dword	0 .. 4294967295
char	Символ, заключенный в единичные кавычки: 'a'.
Real	Вещественные числа записываются в виде знака, мантиссы, десятичной точки, дробной части, символа е и показателя – без пробелов. Знак, дробная часть и показатель необязательны. Примеры чисел: 2.5 -32769 4e-23 Если нужно, целые значения автоматически преобразуются в вещественные.
string	Последовательность латинских букв, цифр и знаков подчеркивания, начинающаяся с символа в нижнем регистре, или последовательность любых символов, заключенных в двойные кавычки. Например: my_name1 "Елена" "123000 Minsk" Строки, которые записываются в тексте программы, не должны быть длиннее 255 символов. Строки, которые вводятся из файла или вычисляются практически имеют неограниченную длину (до 4G).
symbol	Синтаксис такой же, как у строк.

Домены *string* и *symbol* по-разному хранятся и обрабатываются, хотя внешне практически не отличаются. Например, значения типа *symbol* очень быстро сравниваются. При выборе нужно руководствоваться соображениями эффективности программы.

О других возможностях описания доменов будет рассказано позднее.

В **разделе Goal** формулируется назначение программы, т.е. запрос, позволяющий получить ответ системы. Если цель является фактом, то Турбо-Пролог отвечает True или False; если цель содержит переменные, то Турбо-Пролог выдает либо те их значения, которые приводят к решению, если оно существует, либо сообщение No Solution (Нет решений). Например, на запрос

```
Goal:
   likes(X,tennis)    /* Кому нравится теннис?*/
```

система даст ответ следующего вида: X=elena, а на запрос

```
Goal:
   likes(_,tennis)    /* Есть ли кто-нибудь, кому нравится теннис ?*/
```

система даст ответ Yes.

Целевое утверждение может состоять из нескольких частей. В этом случае цель является составной, и каждая часть такой цели называется подцелью. Подцели могут быть сформированы в виде конъюнкции или дизъюнкции.

В разделе constants вы можете дать имена символическим константам. Определение константы имеет вид:

<имя> = <определение костанты>

где <имя> – имя символической константы, <определение константы> – значение константы. На строке может быть только одно описание константы. Перед компиляцией Пролог заменит в программе имена констант на соответствующие строки. Пример раздела констант:

```
constants
   pi = 3.141592653
   path = "c:\\tprolog\\bgi"
   hundred = (10*(10-1)+10)
```

Тогда фрагмент программы на Прологе

```
   … A = hundred * pi …
```

будет обработан компилятором так:

… A = (10*(10-1)+10) * 3.141592653 …

Как работает программа на Прологе. Унификация в Турбо-Прологе

Переменные в Прологе могут быть свободными или связанными. Переменная является свободной, пока она не получила какого либо значения. Переменная, которая имеет значение, называется связанной.

Основой процедуры логического вывода является процесс сопоставления двух термов и присваивания свободным переменным, входящим в них, таких значений, которые сделают эти термы идентичными. Данный процесс называется унификацией. Алгоритм унификации находит необходимые значения переменных, если они существуют, или сообщает об отказе, если таких значений нет. Например, likes(X,tennis) и likes(elena,tennis) унифицируемы при X=elena, а likes(X,tennis) и likes(tom,swimming) не унифицируемы.

При сопоставлении двух термов Турбо-Пролог придерживается следующих правил:

*

одинаковые константы сопоставимы друг с другом;

*

если переменная уже связана, то она действует так же, как обычная константа;

*

свободная переменная сопоставляется с константой или с ранее связанной переменной и становится связанной с соответствующим значением;

*

две свободные переменные связываются друг с другом, с этого момента они трактуются как одна переменная: если одна из них принимает какое-либо значение, то вторая тоже принимает это же значение;

*

свободная переменная может быть связана с составным объектом;

*

два составных терма сопоставимы, когда они имеют один и тот же функтор и одинаковое число аргументов; в этом случае попарно сопоставляются аргументы термов.

В качестве примера унификации составных объектов рассмотрим унификацию объектов birthday(Name,date(_, _, Y)) и birthday(person(“Иван”,”Петров”), date(“Авг.”,2,1980)). В данном случае переменная Name получит значение person(“Иван”,”Петров”), а переменная Y - значение 1980.

Турбо-Пролог выполняет унификацию в двух случаях:

когда цель сопоставляется с заголовком предложения;

когда используется знак равенства, который является инфиксным предикатом (предикатом, который расположен между своими аргументами, а не перед ними).

Например, рассмотрим цель

```
goal
    P1 = birthday(person(“Иван”,”Петров”), date( “Авг.”,2,1980)),
    P1 = birthday(Name,date(_, _, 1980)), write(Name).
```

При согласовании первой подцели переменная P1 получит значение, указанное справа от знака “=“. При согласовании второй подцели P1 уже связана. Так как термы, находящиеся по обе стороны знака “=“ сопоставимы, то переменная Name будет связана со значением person(“Иван”,”Петров”). При согласовании третьей подцели, стандартного предиката write, будет напечатано значение связанной переменной Name.

Поиск с возвратом

В отличие от процедурных языков, таких как Бэйсик или Паскаль, в которых программист должен точно описать процесс решения задачи, Турбо-Пролог является описательным (декларативным) языком. Программа на Прологе содержит описание проблемы, для этого используются правила и факты, и цель. Выполнение программы

заключается в попытке доказать целевое утверждение, используя предположения , заданные в программе. Механизм поиска решений“встроен” в Пролог, он называется “поиск с возвратом”.

Напомним, что каждый факт и каждое правило в программе называется утверждением. Правило - это утверждение, имеющее заголовок и тело. Факт - это утверждение, имеющее заголовок и пустое тело. Совокупность утверждений, заголовки которых имеют одинаковый функтор, называется процедурой.

Основные принципы механизма поиска решений в Прологе:

- Подцели согласуются по порядку, сверху вниз, слева направо.
- Для каждой цели или подцели Пролог просматривает утверждения в том порядке, в котором они появляются в программе.
- Цель считается согласованной, если она сопоставляется с заголовком какого-либо утверждения процедуры. При согласовании цели с процедурой, имеющей пустое тело (факт), цель согласуется немедленно. Если согласование цели происходит с заголовком утверждения, то цель согласуется только тогда, когда каждая подцель в теле этого правила будет согласована после вызова ее в качестве текущей цели.
- Присваивание значений переменным выполняется внутренними программами унификации всякий раз когда свободная переменная сопоставляется с константой. В этом случае переменная получает значение константы. Переменная вновь становится свободной, когда сопоставление оказывается неуспешным.
- В случае если попытка согласования цели оказалась неудачной, Пролог выполняет откат для определения альтернативных путей вычисления цели или подцели. При отыскании решений Пролог предусмотрительно устанавливает невидимые указатели – точки отката в местах, где возможен более чем один путь согласования цели.

Рассмотрим программу Prim0403.pro, которая содержит сведения об именах и возрастах нескольких игроков в теннисном клубе:

```
domains
    child = symbol
    age   = integer
predicates
    player(child, age) - nondeterm (o,i), nondeterm (i,i) /*игрок(ребенок,возраст)*/
clauses
    player(peter, 9).
    player(paul, 10).
    player(chris, 9).
    player(susan, 9).
```

Цель состоит в организации двух игр для каждой пары игроков, возраст которых не превышает 9 лет.

Это целевое утверждение можно сформулировать следующим образом:

```
Goal:
    player(Person1,9),player(Person2,9),Person1<>Person2.
```

На первом шаге Пролог пытается найти решение для первой подцели player(Person1,9). Эта подцель согласуется путем сопоставления Person1 с peter, так как Пролог просматривает утверждения для согласования сверху вниз.

После достижения первой подцели Пролог переходит ко второй подцели: player(Person2,9). Эта подцель опять согласуется при сопоставлении Person2 с peter.

На следующем шаге Пролог переходит к согласованию третьей подцели: Person1<>Person2. Так как Person1 и Person2 имеют значения peter, то данная подцель не согласуется, и Пролог осуществляет откат к предыдущей подцели player(Person2,9). Эта подцель сопоставляется с фактом player(chris,9), и Person2 согласуется с crhis .Теперь Пролог опять в качестве текущей подцели выполняет цель Person1<>Person2 . Эта подцель успешна, так как peter и chris отличны. Таким образом, Пролог согласовал все подцели

целевого утверждения, и цель считается согласованной. Получена первая пара игроков : Peter-Chris.

Однако, Пролог при выполнении внешнего целевого утверждения должен найти все возможные решения целевого утверждения. Для этого он возвращается к предыдущей цели, освобождая переменную Person2. Так как player(Person2,9) согласуется с player(susan,9), то Пролог снова переходит к третьей подцели . Согласование подцели завершается успешно, следовательно второе решение целевого утверждения найдено (пара Peter-Susan).

Для отыскания следующего решения Пролог возвращается ко второй подцели, но так как для этой подцели все варианты исчерпаны, Пролог выполняет откат к первой подцели. Переменная Person1 становится свободной и может быть согласована с chris. Для согласования второй подцели Пролог выполняет поиск сверху вниз, и связывает переменную Person2 с peter. Третья подцель завершается успехом, так как chris и peter различны. Целевое утверждение считается согласованным, и следующая пара игроков Chris-Peter.

В поисках еще одного решения целевого утверждения Турбо-Пролог возвращается ко второй подцели. При выполнении отката переменная Person2 становится свободной и при выполнении унификации связывается с chris. Однако, Person1 и Person2 в этом случае эквивалентны, и третья подцель завершается неуспехом. Пролог опять выполняет поиск с возвратом, и переменная Person2 сопоставляется с susan. Третья подцель выполняется, следовательно, получено четвертое решение для целевого утверждения (пара Chris-Susan).

И вновь для отыскания всех решений Пролог возвращается ко второй подцели, но на этот раз безуспешно. Следовательно, Пролог снова выполняет откат к первой подцели, и свободная переменная Person1 сопоставляется с susan. Далее выполняется согласование второй подцели, и переменная Person2 сопоставляется с peter. Третья подцель успешна, и, следовательно, получено пятое решение (Susan-Peter).

На следующем шаге Пролог выполняет возврат ко второй подцели, и Person2 сопоставляется с chris. Таким образом получена шестая пара: Susan и Chris.

Последнее исследуемое решение завершается неуспехом, так как Person1 и Person2 идентичны.

Лабораторная работа 2
Управление поисковым решением

Может сложиться впечатление, что раз механизм поиска решений “встроен” в Пролог, то им нельзя управлять. На самом деле это не так. В этой работе рассматриваются некоторые методы, которые позволяют управлять работой программы на Прологе.

Отсечение

Для прерывания поиска с возвратом в Прологе используется предикат cut или отсечение (обозначается !). Этот предикат, вычисление которого всегда завершается успешно, помещается в тело правила как любая другая подцель. После того, как процесс прошел через отсечение, уже нельзя выполнить поиск с возвратом в подцелях, расположенных в обрабатываемом предикате перед отсечением, а также нельзя возвратиться к другим утверждениям, определяющим предикат, содержащий отсечение.

Отсечение применяется в двух случаях:

- Когда вы заранее знаете, что поиск альтернативных решений не приведет к разумному результату. Это бесполезная трата времени и памяти. Если вы используете отсечение в этой ситуации, программа будет работать быстрее и использовать меньше памяти. Такие отсечения называются зелеными.
- Когда логика программы нуждается в отсечениях, чтобы предотвратить рассмотрение альтернативных подцелей. Такие отсечения называются красными.

Пример:

Пусть в программе есть сведения о марках автомобилей, их цене и дате выпуска. Определим предикат pr_inf, которые печатает “подходит”, если машина выпущена в 1993 году, и “не подходит” – в противном случае.

```
domains
   name=string          /*марка автомобиля*/
   price,year=integer      /*цена, год выпуска */
predicates
   nondeterm car(name,price,year)- (i,o,o),(o,o,o),(o,o,i)
   pr_inf(year) - multi (i)
clauses
   car("Москвич",8000,1995).
   car("Волга",8000,1995).
   car("ВАЗ 2108",8000,1994).
   car("ВАЗ 2108",7500,1993).
   car("ВАЗ 2109",8000,1993).
   car("Таврия",8000,1992).
   pr_inf(Y):-Y=1993,write(" подходит"),nl.
   pr_inf(_):-write(" не подходит "),nl.
```

Здесь nl - стандартный предикат Пролога. При печати он посылает на экран комбинацию спецсимволов “возврат каретки” и “перевод строки”, то есть переводит курсор в начало следующей строки.

Если в качестве цели задать car("ВАЗ 2109",_,Y),pr_inf(Y), то получится странное решение: будет напечатано “подходит” и “не подходит”. Это произошло из-за того, что найдя нужную машину и напечатав “подходит”, Пролог пытается найти все возможные решения и просматривает оставшиеся альтернативы для pr_inf. Чтобы остановить дальнейший поиск, когда решение найдено, нужно использовать отсечение. Заменим первое правило для pr_inf на:

```
   pr_inf(Y):-Y=1993,!,nl,write(" подходит").
```

В этом случае программа будет работать правильно. Можно было бы поставить отсечение и после nl или write, так как эти подцели не имеют альтернатив. Программа будет работать так же. Если же поставить отсечение перед проверкой Y=1993, то ни для каких значений Y не может быть напечатано "не подходит", так как отсечение запрещает возврат к другим утверждения для pr_inf после того как выяснится, что год выпуска не равен 1993.

Программирование повторяющихся операций

Очень часто в программе необходимо выполнить одну и ту же задачу несколько раз. В Прологе повторяющиеся операции обычно выполняются при помощи определенного способа построения правил.

Существуют два способа реализации правил: повторение и рекурсия. Правила, выполняющие повторения, используют откат, а правила, выполняющие рекурсию, используют самовызов.

При реализации отката в программах применяются различные методы. В частности, в Прологе имеется специальный предикат fail, который всегда вызывает аварийное завершение и, следовательно, приводит к откату. Действие предиката fail равносильно эффекту от сравнения 2=3 или другой невозможной подцели. Обратите внимание, что помещать подцель после fail в теле правила бессмысленно, так как fail всегда заканчивается неудачно.

Покажем, например, как с помощью предиката fail можно найти все возможные решения задачи. Рассмотрим предыдущую программу, добавив к ней предикат pr_car и цель, указанную ниже.

```
predicates
   pr_car
```

```
clauses
    pr_car:-car(Name,_,_),write(Name),nl,fail.
    pr_car.
goal
    pr_car,write("Других автомобилей нет"),nl.
```

Обратите внимание, что для определения предиката pr_car используются два утверждения. Их порядок в данном случае очень важен! Первое правило заканчивается предикатом fail, который всегда вызывает откат. Первый раз переменная Name получит значение "Москвич", которое и будет напечатано. Затем произойдет откат, и при отсутствии альтернатив у nl и write , будет отыскиваться другое решение для предиката car. При этом сначала освобождается переменная Name, затем находится альтернативное решение для предиката car, и переменная Name связывается с новым значением. В конце концов будут просмотрены все альтернативы. Тогда и будет выполнено второе утверждение для предиката pr_car. Если бы его не было, то pr_car всегда завершался бы неудачно, и, следовательно, слова "Других автомобилей нет" никогда бы не напечатались.

Отрицание

Стандартный предикат Турбо-Пролога not (отрицание) позволяет задать в качестве подцели отрицание некоторого утверждения. Его вид

not(<утверждение>)

Этот предикат успешен, если <утверждение> представляет собой цель, оказавшуюся при доказательстве неуспешной.

Утверждение внутри not не может содержать свободных переменных, но может содержать анонимные переменные.

Задание

1. Выполните приведенную выше программу. Разберитесь, как она работает.

Измените предикат pr_car так, чтобы он выводил на экран машины с заданным годом выпуска. В разделе описания предикатов в этом случае его нужно описать, например, так:

multi pr_car(year) - (i), (o)

Испробуйте различные цели: напечатать названия машин, выпущенных в 1995 году; напечатать названия машин, выпущенных в 2000 году; напечатать названия всех машин. Объясните полученный результат.

2. Определите в приведенной ниже программе предикат "обычный студент". Студент обычный, если он имеет средний балл <=4 и не любит лекции.

Используя этот предикат, выведите на экран фамилии необычных студентов.

Указание: используйте предикат not.

```
domains
  name = symbol
  ball  = real

predicates
  nondeterm ordinary_student(name)
  nondeterm student(name,ball) -  (o,o),(i,o)
  nondeterm like_lecture(name)

clauses
  student("Иванов Иван",3.5).
  student("Сидорова Мария",4.5).
  student("Петров Петр",3.9).

  like_lecture("Сидорова Мария").
```

like_lecture("Иванов Иван").

3. Выполните приведенные программы. Объясните, чем они отличаются.

Таблица 3

Вариант 1	Вариант 2
predicates action(integer) - nondeterm (i) clauses action(1):- nl, write("Вы ввели 1."),nl. action(2):- nl, write("Вы ввели два."),nl. action(3):- nl, write("Была введена цифра 3."),nl. action(N):- nl, N<>1, N<>2, N<>3, write("Я не знаю такого числа!"). goal write("Введите число от 1 до 3: "), readint(Num), action(Num).	predicates action(integer) - procedure (i) clauses action(1):-!, nl, write("Вы ввели 1."). action(2):-!, nl, write("Вы ввели два."). action(3):-!, nl, write("Была введена цифра 3."). action(_):-nl, write("Я не знаю такого числа!"). goal write("Введите число от 1 до 3: "), readint(Num), action(Num),nl.

Объясните:

какая программа эффективнее и почему;

важен ли порядок правил для предиката action;

может предикат action завершиться fail;

сколько решений имеет предикат action;

какой тип отсечений мы имеем в данном случае: красные или зеленые.

Лабораторная работа 3
Управление поисковым решением

Использование предиката repeat

Еще один метод, реализующий циклические операции, основан на использовании предиката следующего вида:

repeat.

repeat:-repeat.

Такой предикат обеспечивает бесконечное множество различных решений (так как определен рекурсивно), и позволяет применять поиск с возвратом в тех случаях, когда в задаче нет других альтернативных правил. Можно вместо repeat написать другое имя. Для пояснения использования предиката рассмотрим следующий пример.

Пример:

Ниже приведена программа, в которой вводимые с клавиатуры символы выводятся на экран. Признаком конца ввода является нажатие клавиши Enter.

predicates

```
  repeat - nondeterm ()
  typewriter - nondeterm ()

clauses
  repeat.
  repeat:-repeat.

  typewriter:-
        repeat,
        readchar(C),   /* предикат считывает символ с клавиатуры */
        write(C),
        C = '\r'.      /* сравниваем значение с клавишей Enter */

goal
  write("Вводите символы, в конце нажмите Enter:"),nl,
  typewriter,nl.
```

Рассмотрим подробнее, как работает typewriter. Сначала выполняется repeat, который ничего не делает и всегда завершается успешно. Затем предикат readchar(C) считывает очередной символ и присваивает его переменной C, а предикат write(C) выводит этот символ на экран. Если введенный символ не является клавишей Enter, то подцель C = '\r' завершается неуспешно, и происходит откат к последней точке возврата. Так как readchar и write не имеют альтернатив, то происходит возврат к repeat, который всегда имеет альтернативные решения. Если нажата клавиша Enter, подцель успешна, и предикат завершает работу.

Методы организации рекурсии

Правило, содержащее само себя в качестве компоненты, называется правилом рекурсии. Рекурсивные правила являются одним из способов организации циклических процессов (повторяющихся операций).

Примером рекурсивной процедуры может служить задача считывания символов с клавиатуры. Признаком завершения ввода служит символ #.

Пример:

```
predicates
  write_ch - nondeterm ()
  read_ch - nondeterm ()

clauses
  write_ch:-write("Введите символы" ),nl,
        write("Для завершения введите символ #"),nl,nl.
  read_ch:-readchar(C),
        C<>'#',!,
        write(C),
        read_ch.
  read_ch.

goal
  write_ch, read_ch, nl.
```

Программа циклически считывает символ, введенный пользователем: если этот символ не '#', то он выдается на экран, если этот символ '#', то программа завершается.

Правило рекурсии обязательно должно содержать условие выхода. В противном случае рекурсия бесконечна.

У рекурсии есть один недостаток – всякий раз, когда одна процедура вызывает другую, информация о вызывающей процедуре должна быть сохранена для того, чтобы возобновить выполнение процедуры на том же месте, где она остановилась. Информация сохраняется в стеке, который при большом числе вызовов может переполниться. Поэтому особый интерес вызывает случай, когда после завершения работы вызванной процедуры, вызывающей процедуре ничего не надо делать. В этом случае информацию о вызывающей процедуре можно не сохранять. При каждом новом вызове достаточно присвоить аргументам новые значения. Эта операция называется оптимизацией хвостовой рекурсии или оптимизацией последнего вызова. Она возможна, если выполняются следующие условия:

- рекурсивный вызов является самой последней подцелью предложения;
- ранее в предложении не было возвратных точек.

Задание

Выполните приведенную выше программу. Разберитесь, как она работает. Можно ли в предикате repeat поменять местами правила? Объясните почему.

Измените предикат typewriter так, чтобы при нажатии клавиши печатался не только символ, но и его ASCII-код. Используйте стандартный предикат char_int().

Переделайте программу так, чтобы она заканчивала работу при нажатии клавиши Esc (код клавиши 27).

Лабораторная работа 4
Использование списков в Прологе

Рекурсивными в Прологе могут быть не только утверждения, но и структуры данных. Примером такого типа данных является список.

Список - это рекурсивный составной объект. Он состоит из двух частей: головы, которой является первый элемент, и хвоста, которым является список, включающий все последующие элементы. Следовательно, хвост списка - всегда список (возможно пустой), голова списка - всегда элемент.

В Прологе список заключается в квадратные скобки. Для отделения головы списка от хвоста используется знак вертикальной черты "|". Пустой список обозначается [ ]. Пустой список нельзя разделить на голову и хвост.

Можно использовать упрощенное представление, при котором рекурсивная природа списков оказывается скрытой. В такой записи список изображается в виде последовательности элементов, ограниченной квадратными скобками и разделяемой запятыми. Пустой список, применяемый для завершения рекурсивного описания, явно не указывается. Ниже приведены примеры списков. Записи в разных столбцах эквивалентны.

[a|[ ]]	[a]		
[a|[b|[ ]]]	[a|[b]]	[a,b]	
[a|[b|[c|[ ]]]]	[a|[b|c]]	[a|[b,c]]	[a,b,c]

Элементами списка должны быть объекты одного и того же типа, например, целые числа, вещественные числа, символы, строки, структуры или списки. Количество элементов в списке называется его длиной. Длина списка может быть любой, единственным ограничением является объем оперативной памяти.

Примеры списков (в первом столбце приводится список, во втором - его голова, в третьем - хвост):

[1,2,3,4,5,6,7]	1	[2,3,4,5,6,7]
['a', 'b', 'c']	'a'	['b', 'c']
[[1,2,3], [2,3,4], [ ]]	[1,2,3]	[[2,3,4], [ ]]

["Москва"] "Москва" [ ]
[ ] не определена не определен

Примеры унификации списков (в первых двух столбцах приводятся списки, в третьем – результат сопоставления):

[X, Y, Z] ["Иван", "ест", "мороженое"] X="Иван", Y="ест", Z="мороженое"
[7] [X | Y] X=7, Y=[]
[1, 2, 3, 4] [X, Y | Z] X=1, Y=2, Z=[3,4]
[1, 2] [3 | X] fail

Для описания списка в Прологе используется конструкция вида:

```
domains
   objects=.........
   objectlist=objects*
```

Звездочка при описании домена указывает на то, что данный объект является списком.

Так как список – это объект данных, он используется в Прологе в качестве аргументов предикатов, например, pr_list([1,2,3,4]).

Для выполнения большинства операций над списками, таких как вывод на печать элементов списка, определение принадлежности элемента списку, деление списка на два, соединение списков и т.д., используется разделение списка на хвост и голову. Для соответствующего предиката, как правило, имеются два утверждения, одно из которых является рекурсивным правилом обработки списка, а другое – граничное условие для завершения рекурсивной обработки списков. Для пояснения методов обработки списков рассмотрим несколько примеров.

Пример:

```
          /*Вывод на экран элементов списка */
domains
   list_int=integer*      /* список целых чисел */
predicates
   pr_list(list_int)
clauses
   pr_list([ ]).      /*Граничное условие: если список пуст, то ничего не делать*/
   pr_list( [H|T] ):-write(H),nl,pr_list(T).  /* Рекурсивное правило: если список не пуст,
                        то разделить список на хвост и голову и напечатать голову.*/
goal
   pr_list([4,-9,5,3]).
```

При выполнении данного целевого утверждения Пролог выполняет поиск фактов и правил сверху вниз. Первый вариант правила (граничное условие) дает неуспех, так как его объект является пустым списком. Целевое утверждение унифицируется с головой второго правила pr_list (рекурсивное правило). При этом переменной H присвавается значение первого элемента в списке (4), а переменной T ставится в соответствие оставшаяся часть списка

[-9,5,3]. Следующей подцелью становится предикат write(H). После печати элемента и перевода строки (nl), текущей подцелью становится предикат pr_list(T) . Для удовлетворения текущей подцели Пролог просматривает первое предложение, которое снова не подходит, а затем второе. Переменной H присваивается значение -9, а переменной T - [5,3]. Операция повторятся до тех пор, пока текущая подцель не примет вид: pr_list([]). В этом случае подцель согласуется с первым предложением. Этот вариант не имеет рекурсивных вызовов, что приведет к сворачиванию рекурсии.

Пример 2.

Два варианта предиката, вычисляющего длину списка.

Таблица 5

domains
  list = integer*
predicates
  length_of(list,integer)
clauses
  length_of([], 0).
  length_of([_|T],L):-
  length_of(T,TailLength),
  L = TailLength + 1.
goal
  length_of([1, 2, 3], L).
	domains
	  list = integer*
	predicates
	  length_of(list,integer,integer)
	clauses
	  length_of([], Result, Result).
	  length_of([_|T],Result,Counter):-
	  NewCounter = Counter + 1,
	  length_of(T, Result, NewCounter).
	goal
	  length_of([1, 2, 3], L, 0).

Приведем еще несколько предикатов, которые реализуют различные операции над списками.

```
domains
   list_int=integer*      /* список целых чисел */
predicates
   nondeterm member(integer,list_int) - (i,i), (o,i)
   nondeterm append(list_int,list_int,list_int) - (i,i,o), (o,i,i), (o,o,i)
   sort(list_int,list_int)
   insert(integer,list_int,list_int)
   asc_order(integer,integer)
clauses
         /* Поиск  элемента в списке: первый аргумент - что ищем,
         второй аргумент - список, в котором ищем */
   member(X,[X|_]).
   member(X, [_|T] ):- member(X,T).
         /* Слияние списков: все три аргумента - списки, третий список - результат
         слияния первых двух списков */
   append([],L,L).
   append([N|L1],L2,[N|L3]):- append(L1,L2,L3).
         /* Сортировка списков: первый аргумент - исходный список,
         второй аргумент - отсортированный по возрастанию список.
         Реализована сортировка вставкой.*/
   sort([],[]).
   sort([X|T],L):-sort(T,L1), insert(X,L1,L).
   insert(X,[Y|L],[Y|L1]):-asc_order(X,Y),!, insert(X,L,L1). /* вставляет
         первый аргумент в список, заданный в качестве второго аргумента,
         результат - третий аргумент */
   insert(X,L,[X|L]).
   asc_order(X,Y):-X>Y. /* вспомогательный предикат */
```

Если в качестве цели задать member(2,[1,2,3]), то Пролог ответит Yes (цель доказана). Для цели append([1,2],[3],X), будет найдено решение X=[1,2,3], а для цели sort([2,5,1,4],X) - решение X=[1,2,4,5]. Однако, это еще не все. Зададим в качестве цели member(X,[1,2,3]), и Пролог найдет 3 решения: X=1, X=2 и X=3. Аналогично, если в качестве цели задать append(X,[3],[1,2,3]), будет найдено решение X=[1,2].

В Прологе нередко предикат, созданный для решения одной задачи, может решать и другие. Состояние аргументов при вызове предиката называется потоком параметров. Аргумент, который имеет значение в момент вызова, называется входным и обозначается буквой (i), а свободный аргумент - это выходной аргумент, обозначается буквой (o). Потоки параметров для стандартных предикатов обычно указываются в справочных руководствах.

В практических программах часто бывает необходимо собрать данные в список для последующей их обработки. С этой целью используется встроенный предикат

findall(Var, Predicat, List_name),

где Var (имя переменной) обозначает объект, который необходимо включить в список, Predicat определяет предикат, из которого нужно собрать значения, а List_name - это имя переменной выходного списка. Переменная Var должна являться одним из аргументов предиката Predicat. Например, Вы имеете предикат person(name, age) (человек(имя,возраст)) и факты:

person(tom,26).
person(bill,28).
person(sue,20).

Предположим, что необходимо определить список всех имен, имеющихся в базе данных. Эта операция при использовании встроенного предиката findall будет выглядеть следующим образом:

findall(Name,person(Name,_),Name_list).

После выполнения findall переменная Name_list получит значение [tom,bill,sue].

Задание

1.Проанализируйте приведенные выше предикаты. Разберитесь, как они работают.
Можно ли поменять местами правила для предиката member? Попробуйте задать цель member(X, [1, 2, 3, 4, 5]) в обоих случаях.

Для предиката append испробуйте цели:

append([1, 2, 3], [5, 6], L).

append([1, 2], [3], L), append(L, L, LL).

append(L1, L2, [1, 2, 4]).

append(L1, [3,4], [1,2,3,4]).

2.Напишите предикаты, которые выполняют следующие действия:

- вычисляет сумму элементов списка;
- добавляет ко всем элементам списка 1;
- удаляет отрицательные элементы из списка.

4. Определите в своей программе предикат person, содержащий сведения о фамилии, месте жительства (город) и возрасте. Задайте несколько фактов. Напишите программу, которая по имеющимся фактам определит средний возраст жителей указанного города.

Лабораторная работа 5
Базы данных в Прологе

Средства, которые имеются в Прологе для организации баз данных (БД), рассчитаны на работу с реляционными БД, так как Пролог можно использовать в качестве мощного языка запросов именно для реляционных БД. Внутренние унификационные процедуры языка автоматически выбирают факты с нужными значениями известных параметров и присваивают значения неизвестным параметрам. Алгоритм поиска с возвратом позволяет находить все решения для сделанного запроса.

База данных называется динамической, если в процессе работы в нее можно добавлять новые утверждения и удалять любые содержащиеся в ней утверждения. Динамическая БД может быть записана на диск и считана с диска в оперативную память. Статические БД не могут быть изменены в процессе выполнения программы, так как утверждения такой БД являются частью программы.

Факты и правила, содержащиеся в разделе clauses программы на Прологе, можно рассматривать как статическую БД. Динамическими в Прологе являются внутренние и внешние БД.

Внутренняя БД состоит из фактов, которые можно добавлять и удалять во время выполнения программы.

Создание внутренних баз данных

Внутренняя БД Пролога может содержать только факты (не правила!). Факты, принадлежащие внутренней БД, используются как обычные предикаты. Как и другие предикаты они должны быть объявлены до их использования, но не в разделе predicates, а в специальном разделе facts или database. Например:

```
database
    db_person(name,age,addres)
```

Можно при описании указать имя БД, отделив его от database знаком "-". Если имя не задано, БД получит стандартное имя dbasedom. Допускается наличие нескольких секций database, но тогда необходимо указать имя каждой секции (за исключением быть может одной). Это позволяет сохранять и загружать секции базы данных отдельно. В двух различных секциях database нельзя использовать одинаковые имена предикатов.

```
database - mydatabase
    likes(simbol,symbol)
database - people
    person(string,string)
```

Факты для предикатов, описанных в разделе database (facts), во время разработки программы могут быть помещены в раздел clauses как обычно. Во время выполнения программы их можно удалить так же, как и добавленные позднее факты.

По своей природе предикаты БД недетерминированные. Так как факты могут быть добавлены во время выполнения программы, компилятор всегда предполагает наличие альтернативного решения при поиске с возвратом. Если у вас имеется предикат, для которого в БД никогда не будет более одного факта, то его можно описать как determ. В этом случае к существующему факту вы не сможете добавить ни одного факта, возникнет ошибка.

С фактами внутренней БД можно работать как с термами, так как по описанию БД Пролог создает домен с именем, совпадающим с именем секции database (или с именем dbasedom, если секция БД не имела имени). Для каждого предиката БД создается одна альтернатива. Этот домен может быть использован как любой другой домен. Для объявленной выше БД people будет создан домен people = person(string, string). Домен people может использоваться как любой другой домен. Например, создадим предикат no_copy, который не позволит добавить в БД факт, если там такой факт уже есть:

```
predicates
    no_copy (people)
clauses
    no_copy (person(Name,Address)):-
        person(Name,Address), ! ;
        assert(person(Name,Address)).
```

Чтобы добавить в БД новые факты во время выполнения программы следует использовать стандартные предикаты

```
asserta(<факт>)                    /* (i)   */
asserta(<факт>,databaseName)       /* (i,i)*/
assertz(<факт>)                    /* (i)   */
assertz(<факт>,databaseName)       /* (i,i)*/
assert(<факт>)                 /* (i)  */
```

assert (<факт>,databaseName) /* (i,i)*/

Предикат asserta вставляет новый факт в БД перед имеющимися фактами для данного предиката, а assertz и assert помещают новые утверждения вслед за имеющимися в БД утверждениями того же предиката. Например, assert(person("Михаил","111-11-11")) добавит факт person("Михаил","111-11-11") в конец БД. Факты, добавляемые в БД, не могут содержать свободных переменных.

Для удаления предикатов из БД используются стандартные предикаты

retract(<факт>) /* (i) */

retract(<факт>,databaseName) /* (i,i)*/

retractall(<факт>) /* (i) */

retractall(<факт>,databaseName) /* (i,i)*/

Предикат retract удаляет первый факт вашей БД, совпадающий с фактом <факт>, а retractall удаляет все утверждения, совпадающие с образцом <факт>. Например, предикат retractall(person("Михаил",_)) удалит из БД все факты person для Михаила, а retract(person("Михаил",_)) удалит первый факт для Михаила. Цель retractall(_,people) удалит все факты БД people.

Во всех вышеприведенных предикатах второй аргумент (имя БД) используется только для проверки типа, поэтому его можно опускать.

Для сохранения в файле БД, созданной в оперативной памяти, применяются предикаты

save(fileName) /* (i) */

save(fileName,databaseName) /* (i,i)*/

При вызове предиката с одним аргументом сохраняются факты из БД dbasedom, при вызове с двумя аргументами - факты из указанной БД. Например, предикат save("d:\\prolog\\work\\bd.dba",mydatabase) сохраняет факты БД mydatabase в файле bd.dba.

Для использования базы данных, сохраненной в файле, ее можно загрузить в оперативную память с помощью предикатов

consult(fileName) /* (i) */

consult(fileName, databaseName) /* (i,i)*/

Предикат consult считывает из файла факты и вставляет их в конец соответствующей БД. Если предикат вызван с одним параметром, то будут считаны факты, которые были описаны в секции без имени. Например, consult("d:\\prolog\\work\\bd.dba", mydatabase) загрузит из файла bd.dba факты БД mydatabase. Если файл содержит что-то кроме фактов указанной БД, то выдается сообщение об ошибке. Факты считываются по одному, поэтому, если из десяти фактов в шестом была ошибка, то первые 5 фактов будут занесены в БД. Предикат consult считывает факты только в том формате, в котором их записывает предикат save. Файлы не должны содержать:

- прописных символов;
- пробелов(кроме строк внутри "");
- комментариев;
- пустых строк;
- символов без двойных кавычек.

Задание

1.Выполните приведенную ниже программу. Это пример простейшей экспертной системы. Разберитесь, как она работает.

```
domains
  thing = string
  conds = cond*
  cond  = string
database  /*facts*/
  is_a(thing,thing,conds)
```

```
  type_of(thing,thing,conds)
  false(cond)

predicates
  run(thing) - nondeterm (i)
  ask(conds) - nondeterm (i)
  update - procedure ()

clauses
  run(Item):-
  is_a(X,Item,List),
  ask(List),
  type_of(ANS,X,List2),
  ask(List2),
  write( Item,", который вам нужен, - это ", Ans),nl.

  run(_):-
  write("Не хватает данных "),
  write("для получения выводов."),
  nl.

  ask([]).
  ask([H|T]):-
  not(false(H)),
  write("Это используется для "),
  write(H," (нажмите y/n)"),
  readchar(Ans), nl, Ans='y',
  ask(T).

  ask([H|_]):-
  assertz(false(H)), fail.

  is_a("язык","Инструмент",["общения"]).
  is_a("молоток","Инструмент",["постройки дома","ремонта забора",
                              "колки орехов"]).
  is_a("швейная машина","Инструмент",["изготовления одежды",
                                      "ремонта парусов"]).
  is_a("плуг","Инструмент",["пахоты","обработки земли"]).

  type_of("english","язык",["общения с людьми"]).
  type_of("prolog","язык",["общения с компьютером"]).
  type_of("молоток","молоток",[]).
  type_of("швейная машина","швейная машина",[]).
  type_of("плуг","плуг",[]).

  update:-
  retractall(type_of("prolog","язык",["общения с компьютером"])),
  asserta(type_of("PDC Prolog","язык",
                          ["общения с персональным компьютером"])),
  asserta(type_of("prolog","язык",
                          ["общения с универсальным компьютером "])).
```

```
goal
  run("Инструмент").
```

2.Сохраните базу данных в файле. Просмотрите полученный файл, используя, например, редактор WordPad.
3.Измените цель, добавив перед предикатом run предикат update. Выполните программу. Просмотрите полученную в результате БД.
4.Удалите из текста программы факты БД is_a и type_of (или закомментируйте их). Измените цель так, чтобы БД сначала загружалась из файла, затем выполнялся предикат run и БД вновь сохранялась на диске.
5.Объясните назначение фактов false. Предусмотрите удаление всех фактов false из БД после ее загрузки с диска.
6.Напишите программу которая позволяет работать с БД «Телефоны». Примерный текст приведен ниже. Допишите предикаты, которые будут добавлять информацию в БД, удалять информацию об абонентах из БД и выдавать телефоны указанных абонентов.

```
database
dbperson(name,phon)

predicates
nondeterm repeat
new_db

clauses
repeat.
repeat:-repeat.

new_db:-
  repeat,
  write("Введите имя => "),
  readln(Name) ,
  write("Введите телефон => "),
  readln(Phon),
  assertz(dbperson(Name,Phon)),
  write(Name," добавлен в БД"),nl,
  write("Будете еще вводить данные? (нажмите y/n)"),
  readchar(Ans), nl, Ans='n',!,
  save("c:\\wwww\\my_db").

goal
  new_db.
```

Лабораторная работа 6
Создание приложения

Для создания новых приложений в Прологе используется **Application Expert**. Чтобы его открыть выполните команду меню **Project | New Project**. Проверьте, чтобы на вкладке **Target** были установлены следующие значения: Platform – Windows32, UI Strategy – VPI, Target Type – exe, Main Program – Prolog (они должны быть установлены по умолчанию).

На вкладке **General** нужно указать имя проекта (оно одновременно является именем файла и именем предиката, поэтому должно удовлетворять соответствующим требованиям) и каталог, в котором будет храниться проект. Рекомендуется хранить каждый проект в отдельном каталоге. Если указан несуществующий каталог, то он будет создан. Для

просмотра дерева каталогов нажмите кнопку **Browse**. Если включить опцию Multiprogrammer Mode, все компоненты проекта будут сохраняться в отдельных файлах, а также будет сгенерирован .PRJ–файл и обеспечена соответствующая поддержка для работы с ним. Эту опцию можно изменить на любой стадии разработки проекта. Чтобы изменять установки в существующем проекте, нужно вновь открыть **Application Expert**, выполнив команду **Options | Project | Application Expert**.

На вкладке **VPI Options** нужно отметить те средства, которые требуется включить в проект. Все эти установки за исключением "toolbar and the help line" позднее можно изменить. Панель инструментов также можно добавить позже, но код в этом случае придется добавлять вручную, используя **Toolbar Editor** и **Toolbar Expert**.

После того, как все опции будут установлены, нажмите на кнопку **Create**. Окно проекта будет содержать 2 модуля VPITOOLS.PRO и MYPROJ.PRO (если вы указали имя проекта MYPROJ). MYPROJ.PRO – главный модуль. Файл VPITOOLS.PRO включает средства, выбранные в **Application Expert**. В главном модуле имеются следующие разделы:

- управление событиями для окна задач (Task window – это главное окно приложения, которое создается при запуске приложения и уничтожается при завершении работы приложения);
- главная цель проекта;
- создание панели инструментов и строки статуса в проекте;
- создание окна About.

Чтобы откомпилировать и выполнить созданный проект выполните команду меню **Project | Run**.

Все компоненты вашего приложения регистрируются в окне проекта. Кнопки с левой стороны окна позволяют выбрать нужный компонент. Кнопки с правой стороны предназначены для выполнения действий над этими компонентами. Используя кнопку **Module** (слева) и **Edit** (справа) просмотрите содержимое созданных модулей.

Изменение меню в Menu Editor

В окне проекта нажмите на кнопку **Menu** на левой панели инструментов. Далее выполните двойной щелчок на Task Menu (главное меню приложения), чтобы активизировать редактор меню. Чтобы добавить новый пункт в меню, передвиньте курсор в нужное место и нажмите на кнопку **New**. В поле **Text** укажите название пункта меню, которое будет выводиться на экране. Можно использовать русские буквы. Наличие '&' перед символом означает, что этот символ используется для быстрого доступа к пункту меню. В поле **Constant** укажите имя пункта меню, которое будет использоваться в программе. По умолчанию это имя начинается с id_ (нельзя использовать русские буквы). Можно также задать комбинацию горячих клавиш и начальное состояние меню. Для создания подменю сначала нажмите кнопку **Submenu**, а затем кнопку **New**. Для возврата в меню более высокого уровня нажмите кнопку **Back**. Если нажать на кнопку **Test**, стандартное меню будет заменено на созданное вами, и вы можете протестировать его. Повторное нажатие кнопки **Test** вернет стандартное меню.

Добавим код, который нужно выполнить при выборе добавленного пункта меню. Для этого сначала в окне проекта выберите **Window**, а затем в появившемся списке выберите Task window и нажмите на кнопку **Code Expert**. На экране появится окно **Dialog and Window Expert**. После этого выберите в списке **Event Type** элемент Menu, а в списке **Event or Item** – имя нужного пункта меню. Нажмите на кнопку **Add Clause**, чтобы добавить в программу обработку нужного события. Название кнопки изменится на **Edit Clause**. Нажмите на эту кнопку, чтобы увидеть текст, добавленный в программу. Если пункт меню назывался id_text, то вы увидите текст:

```
%BEGIN Task Window, id_text

  task_win_eh(_Win,e_Menu(id_text,_ShiftCtlAlt),0):-!,
```

```
    !.
%END Task Window, id_text
```

Комментарии %Begin - %End используются для локализации кода, относящегося к выбранной компоненте интерфейса (в нашем случае пункта меню id_text). Между двумя отсечениями вручную или с помощью мастера добавьте код, который нужно выполнить.

Как создать новое окно

Нажмите на кнопку **Window** в левой части окна проекта, а затем на кнопку **New** в правой части окна. Введите имя окна (в поле **Name**) и заголовок (в поле **Title**). Нажмите **OK**. В этот момент появятся панели, позволяющие добавлять элементы в окно и изменять его размер. Если вы не хотите ничего добавлять в это окно, то просто закройте его.

Создадим новый модуль, в который в дальнейшем поместим код для окна. В окне проекта нажмите кнопку **Module** в левой части окна, а затем на кнопку **New** в правой части окна. Введите имя файла (например, MyWinFile). Выберите из списка тип файла *.pro. Нажмите на кнопку **Открыть**. Появится диалоговое окно **File Inclusions for Module**. Не меняя установок в этом окне, нажмите **OK**.

Далее нужно сгенерировать код для этого окна. Этот код выполняет две функции: создает окно с заданными атрибутами и содержит предикаты, управляемые событиями, то есть обеспечивает реакцию окна на события. Выделите нужное окно в окне проекта и нажмите кнопку **Code Expert**. В поле **Module** выберите созданный модуль (MyWinFile.pro) и нажмите на кнопку **Default Code**. Нажав на кнопку **Edit Code**, вы увидите, какой код был добавлен в указанный модуль.

Задание

1.Создайте новое приложение с опциями, установленными по умолчанию. Запустите его. Проанализируйте созданные файлы. Измените название главного окна приложения. Для этого нажмите на кнопку **Window**, затем на кнопку **Attribute** и в поле Window Title введите новое название окна, например, "Приложение на Прологе".

2.Добавьте в существующее меню приложения новое меню **Тест**, содержащее пункт **Окно сообщения**. При выборе этого пункта меню на экран выводится стандартное окно сообщения с заданным текстом.

Как добавить пункт меню, описано выше. Чтобы вывести окно сообщения, поместите курсор в точку, в которую вы хотите вставить код, в нашем случае перед вторым отсечением, и выполните **Edit | Insert | Predicate Call | Window, Dialog or Toolbar**. В появившемся окне выберите **Common Dialog**, в списке найдите **dlg_Note** и наберите текст сообщения, включая кавычки, например, "Привет !". В результате код будет иметь вид:

```
%BEGIN Task Window, id_text
 task_win_eh(_Win,e_Menu(id_text,_ShiftCtlAlt),0):-!,
    Title="Title",
    dlg_Note(Title,"Привет !"),
    !.
%END Task Window, id_text
```

Можете изменить текст заголовка окна сообщения. Проверьте работу программы.

3.Создайте новое пустое окно. Как это сделать описано выше.

Для проверки работы окна добавьте в меню **Тест** новый пункт **Моё окно** (id_mywin). Чтобы вывести окно на экран, поместите курсор в точку, в которую нужно вставить код, и

выполните **Edit | Insert | Predicate Call | Window, Dialog or Toolbar**. В появившемся окне отметьте **User Defined Window** и в списке выберите нужное окно. Нажмите **OK**. В текст будет вставлен предикат, создающий ваше окно. Запустите приложение и проверьте работу программы.

4.В модуле MyWinFile.pro вставьте следующее правило (3 строки) для предиката win_mywin_eh перед другими правилами для этого предиката:

```
win_mywin_eh(Win, Event, 0):-
    write(Win, ":", Event),nl,
    fail.
```

В результате в окне сообщений будут печататься все события, генерируемые для вашего окна. Запустите программу и попробуйте выполнить различные операции с окном: создать, закрыть, изменить размер, переместить, свернуть, развернуть и т.д.

5.Добавьте в программу код обработки события ***e_Update*** для вашего окна. Оно возникает при перерисовывании окна, например, когда окно или его часть были закрыты другим окном, а потом окно стало активным. Для этого выберите в окне проекта нужное окно и нажмите **Code Expert**. Должны быть выбраны следующие значения: Window – MyWin, Module – MyWinFile.pro, Event Type – Window, Event or Item – e_Update. Нажмите на кнопку **Add Clause**, а затем на кнопку **Edit Clause**, чтобы просмотреть полученный код. Добавьте необходимые предикаты, чтобы правило имело вид:

```
%BEGIN MyWin, e_Update
  win_mywin_eh(_Win,e_Update(_UpdateRct),0):-!,
  RCT = win_GetClientRect(_Win),
  RCT = rct(_,_,R,B),
  draw_Line(_Win, pnt(0,0),pnt(R,B)),
  draw_Line(_Win, pnt(0,B),pnt(R,0)),
  !.
%END MyWin, e_Update
```

В данном случае предикат GetClientRect позволяет получить внутреннюю (рабочую) область окна. Предикаты draw_Line рисуют на поверхности окна две прямые линии в виде креста.

Запустите ваше приложение, откройте два окна и поэкспериментируйте. Вы заметите, что перерисовывается не всё окно, а только та часть, которая была закрыта. Поэтому изображение на экране не всегда корректно. Чтобы устранить этот дефект, нужно несколько изменить правило (оно уже существует) для обработки события e_Size, возникающего при изменении размеров окна, на:

```
%BEGIN MyWin, e_Size
win_mywin_eh(_Win,e_Size(_Width,_Height),0):-!,
  win_Invalidate(_Win),        %добавили эту строчку
ifdef use_tbar
  toolbar_Resize(_Win),
enddef
  !.
%END MyWin, e_Size
```

6.Чтобы напечатать текст в вашем окне нужно использовать предикат draw_Text. Например, draw_Text(_Win, 55, 20, "Мой текст"). Добавьте в окно вывод текста. Чтобы вставлять в текст программы стандартные предикаты удобно использовать комбинацию клавиш Ctrl+Shift+V. Попробуйте еще что-нибудь нарисовать в вашем окне, используя различные предикаты draw_… Для изменения цвета текста используются предикаты:

```
Col=vpi_ComposeRGB(0, 255, 0),    %выбираем цвет
win_SetForeColor(_Win, Col),      %устанавливаем его в качестве цвета текста
```

Цвет фона изменяется аналогично.

Лабораторная работа 7
Использование файлов в Прологе

Использование файлов в Прологе требует описания файлового домена в разделе domains. В отличии от доменов других типов , тип домена file задается по левую сторону от знака равенства,а имя домена - по правую сторону,например:

```
domains
file=myfile;myfile1;myfile2
```

Логическое или символическое имя файла (в данном случае myfile) отождествляется с именем файла в DOS. При использовании в программе нескольких символических имен файлов,имена их доменов указываются через знак (;).

Запись в файл и модификация файла

Перед записью информации в файл его необходимо открыть с помощью предиката

openwrite(SymbolicFileName,DosFileName).

В случае дозаписи информации в файл его следует открыть, используя предикат

oppenappend(SymbolicFileName,DosFileName).

При модификации файла следует использовать предикат

opermodify(SymbolicFileName,DosFileName).

Предикаты openwrite(), openappend() и openmodify() устанавливают связь между пользовательским доменом (myfile) и реальным именем файла в DOS ("File.dat").

Следующая процедура, которую следует выполнить, это назначить файл в качестве устройства записи, используя предикат

writedevice(SymbolicFileName).

Запись информации в файл выполняется с помощью стандартных предикатов

write() и writef().

После записи в файл информации возможно потребуется переназначение устройства записи, в качестве которых могут выступать экран монитора, печатающее устройство или другой файл.

После завершения работы с файлом, его следует закрыть. Для этой цели служит предикат

closefile(SymbolicFileName).

Чтение из файла

Для чтения информации из файла необходимо:

- выполнить открытие файла для чтения , используя предикат

openread(SymbolicFileName);

- назначить файл устройством чтения при помощи предиката

readdevice(SymbolicFile);

закыть файл после завершение чтения из него информации, применив предикат closefile(SymbolicFile).

Рассмотрим пример:

```
domains
file=myfile
predicates
start
read_info(char)
goal
start
clauses
start:- write("Ввод символов с клавиатуры и запись в файл"),
```

```
write("ДЛя завершения ввода нажмите #") ,
openwrite(myfile,"File.dat"),
readchar(X),
read_info(X),
closes(myfile),
write(screen),
write("Запись выполнена").
read_info('#'):-!.    /* Первое правило*/
read_info('\13'):-!,  /* Второе правило*/
write("\13\10"),
writedevice(screen),
write("\13\10"),
readchar(X),
read_info(X).
read_info(X):- writedevice(myfile),/* Третье правило*/
write(X),
writedevice(screen),
write(X),
readchar(Y),
read_info(Y).
```

Данная программа последовательно выполняет следующие подцели:

- вывод на экран сообщения;
- открытие файла для записи;
- считывание символа с клавиатуры;
- вызов правила read_info(X);
- закрытие файла;
- назначение экрана устройством вывода;
- на экран текстового сообщения.

Правило read_info(X) имеет три вырианта. Первый вариант :

read_info('#'):-!

выполняется успешно, если был введен символ '#'.Отсечение с теле этого предложения позволяет исключить поиск других решений.

Второе правило анализируется Прологом, если введенный символ - "возврат каретки", код которого равен 13. После унификации целевого утверждения read_info(X) с головой этого правила выполняются следующие подцели:

- назначается устройство вывода информации (файл);
- в файл записывается последовательность символов "\13\10", что соответствует операции "возврат каретки и перевод на новую сторку";
- назначается устройство вывода информации(экран);
- на экран выводится операция " возврат каретки и перевод на новую строку";
- выполняется считывание с клавиатуры очередного символа;
- выполняется рекурсивный вызов процедуры read_info(X) с новым значением символа;

третье правило read_info(X) выполняется всякий раз, когда введенный символ не является символом '#' или '\13'.

Задание

Создайте окно со следующими свойствами:

- окно должно иметь свое меню;

- окно должно иметь панель инструментов с правой стороны окна, с помощью которой окно может быть очищено;
- окно должно иметь контекстное меню, с помощью которого его можно свернуть, максимизировать, вернуть в нормальное состояние, а также очистить и изменить цвет;
- для создания изображения используется мышь, рисование производится при нажатой кнопке.

1. Создайте новое приложение. Поместим создаваемое окно в отдельный модуль. Выберите **Module**, нажмите на кнопку **New** и введите имя модуля Mywin. Будет создан модуль MYWIN.PRO, в который будет помещен нужный код.
2. Чтобы создать новое меню, вы должны выбрать **Menu** в окне проекта и нажать на кнопку **New**. Дайте новому меню имя 'Меню окна', идентификационная константа id_menu. Нажмите **OK**, чтобы закрыть диалог. Используя **Menu Editor** (вызывается кнопкой **Edit**), добавьте в меню пункт 'Очистить'. Измените для него константу на id_clear.
3. Чтобы создать новое изображение, нажмите в окне проекта на кнопку **Bitmap**, а затем на кнопку **New**. Введите имя (mybmp_norm) и размеры изображения (оставьте по умолчанию). Идентификационная константа и имя файла будут сгенерированы по умолчанию, если вы их не зададите. Нарисуйте небольшое изображение, которое будет использовано в качестве рисунка для кнопки, и закройте графический редактор. Затем создайте ещё одно изображение (mybmp_down), которое будет использовано для нажатой версии кнопки. Скопируйте рисунок (например, с помощью контекстного меню) и измените в нем цвета, например, используя кнопку .
4. Создайте панель инструментов. Выберите в окне проекта **Toolbar** и нажмите на кнопку **New**. Дайте новой панели инструментов имя 'Новая'. После того как вы нажмете **OK**, имя новой панели инструментов появится в списке и появится окно **Toolbar Editor** с его панелью инструментов. Добавьте на панель инструментов кнопку. Укажите имя константы id_clear, Released Bitmap Name – mybmp_norm, остальные два рисунка – mybmp_down. Имя константы совпадает с константой для пункта меню, поэтому не потребуется создавать отдельный код. Используя кнопку **Attribute**, измените значение в поле Toolbar Style так, чтобы панель располагалась справа.
5. Создадим окно. Выберите в окне проекта **Window** и нажмите на кнопку **New**. Укажите имя окна (new_win), меню (Меню окна) и нажмите OK.
6. Панель инструментов (Новая) можно добавить только после того, как окно уже будет создано. Используйте кнопку **Attribute**, а затем кнопку **Toolbars...** . В списке Remaining Toolbars выберите Новая и нажмите на кнопку **Add**. В списке Toolbar также выберите Новая, а в поле Module укажите Mywin.pro. Нажмите на кнопку **Default code**, чтобы сгенерировать необходимый код.
7. Далее нужно создать код для окна. Откройте Dialog and Window Expert (например, с помощью кнопки **Code Expert**). Укажите нужное окно (new_win), модуль, содержащий код, – Mywin.pro. Затем нажмите на кнопку **Default Code**. Для вывода окна на экран добавьте новый пункт в главное меню. Проверьте, как выводится окно на экран. Обратите внимание, что меню окна выводится вместо главного меню.
8. Реализуйте рисование в окне при нажатой кнопке мыши. При обработке событий мыши потребуется БД, в которой будет храниться информация о позиции мыши, состоянии кнопки мыши и указателе на окно. Поместите приведенные ниже строки в файл Mywin.pro после операторов include:

```
DATABASE - mouse
point(WINDOW,PNT)
mouse_isdown(WINDOW)
```

Дальше нужно обработать 3 события мыши: e_MouseDown, e_MouseMove, e_MouseUp. Default code для них нужно добавить, используя Dialog and Window Expert. Окончательный код приведен ниже:

```
win_new_win_eh(_Win,e_MouseUp(_PNT,_ShiftCtlAlt,_Button),0):-!,
    retractall(mouse_isdown(_Win)),
    retractall(point(_Win,_)),
    !.

win_new_win_eh(_Win,e_MouseMove(_PNT,_ShiftCtlAlt,_Button),0):-!,
    mouse_isdown(_Win),
    point(_Win,_PNT1),
    draw_Line(_Win,_PNT1,_PNT),
    retract(point(_Win,_PNT1)),
    assert(point(_Win,_PNT)),
    !.

win_new_win_eh(_Win,e_MouseDown(_PNT,_ShiftCtlAlt,_Button),0):-!,
    assert(mouse_isdown(_Win)),
    assert(point(_Win,_PNT)),
    !.
```

При нажатии на кнопку панели инструментов возникает событие e_Menu для пункта id_clear. Добавьте в программу код, реализующий очистку окна:

```
win_new_win_eh(_Win,e_Menu(id_clear,_ShiftCtlAlt),0):-!,
    win_Clear(_Win, color_White),
    win_Invalidate(_Win),
    !.
```

10. Создайте всплывающее меню. Для этого создайте в редакторе меню как обычно меню, имеющее следующие 4 пункта: Очистить(id_clear), Свернуть(id_close), Максимум(id_max), Нормальное(id_restore). Для пункта Очистить используйте ту же константу (id_clear), что в меню окна и для кнопки панели инструментов.

Добавьте еще одно правило для события e_MouseDown, поместив его перед старым или изменив первую строку старого правила так, чтобы оно выполнялось только для левой кнопки мыши:

```
win_new_win_eh(_Win,e_MouseDown(PNT, _,mouse_button_right),0):-!,
   menu_PopUp(_Win,res_menu(id_popup),PNT, align_Right).

win_new_win_eh(_Win,e_MouseDown(_PNT,_, mouse_button_left),0):-!,
   …
```

Здесь id_popup – идентификационная константа всплывающего меню. У вас она может быть другой.

При создании кода для трех последних пунктов меню, вы не найдете их среди объектов окна в окне Dialog and Window Expert, так как это контекстное меню. Код нужно добавлять вручную. Поместите курсор, например, после предиката обработки события e_Menu для пункта меню id_clear. Введите приведенные ниже предикаты:

```
win_new_win_eh(_Win,e_Menu(id_close,_ShiftCtlAlt),0):-!,
   win_Destroy(_Win).
win_new_win_eh(_Win,e_Menu(id_max,_ShiftCtlAlt),0):-!,
   win_SetState(_Win, [wsf_Maximized]).
win_new_win_eh(_Win,e_Menu(id_restore,_ShiftCtlAlt),0):-!,
   win_SetState(_Win, [wsf_Restored]).
```

Добавим во всплывающее меню еще один пункт, позволяющий изменять цвет линий. Для выбора цвета используется стандартное диалоговое окно. Если идентификационная константа для этого пункта id_color, то соответствущее правило имеет вид:

```
win_new_win_eh(_Win,e_Menu(id_color,_ShiftCtlAlt),0):-!,
   Pen=win_GetPen(_Win),
```

```
Pen = pen(Penwidth,PenStyle,Color),
NewColor = dlg_ChooseColor(Color),
win_SetPen(_Win,pen(Penwidth,PenStyle,NewColor)).
```


5.4 Самостоятельная работа

№ п/п	Тема	Кол. часов	Форма отчетности
1	Информационные системы при интегрированном автоматизированном управлении экономическими объектами	4	1.Опрос на практическом занятии.
2	Тенденции развития информационных систем поддержки решений	4	1.Конспект. 2.Опрос на практическом занятии.
3	Интеллектуальные технологии на основе инженерии знаний и искусственного интеллекта	4	1.Конспект. 2.Опрос на практическом занятии.
4	Экспертные системы - системы, базирующиеся на знаниях	4	1.Конспект. 2.Выполнение самостоятельной работы.
5	Представление знаний в интеллектуальных системах	4	Опрос на практических занятиях.
6	Обработка знаний и вывод решений в интеллектуальных системах	4	Опрос на практическом занятии.
7	Работа инженера знаний при разработке интеллектуальных систем	4	Конспект.
8	Архитектура интеллектуальных информационных систем.	4	Конспект.
9	Инструментальные средства проектирования интеллектуальных систем	4	1.Конспект. 2.Опрос на практическом занятии.
10	Интеллектуальная система планирования производства	4	Конспект.
11	Динамическая интеллектуальная система оперативно- диспетчерского управления предприятием.	4	1.Конспект. 2.Выполнение самостоятельной работы.
12	Информационно-аналитические системы управления маркетингом.	3	Опрос на практическом занятии.
13	Разработка интеллектуальной системы поддержки банковских решений	3	Конспект.
14	Экспертная система риск-менеджмента.	3	Конспект.
15	Основные понятия языка Пролог. Лабораторные работы «Программирование на языке Пролог».	3	1.Конспект. 2.Опрос на практическом занятии.
	Всего по дисциплине	56	
	Форма итогового контроля	Зачет с оценкой	

6 Тематика рефератов, эссе

1. Финансовый анализ и финансовое планирование предприятия с использованием информационно-управляющих систем.
2. Разработка подсистемы оценки эффективности инвестиционных проектов в автоматизированной интеллектуальной системе управления банком.
3. Использование методов оценки эффективности инвестиционных проектов в модели эвристического анализа.
4. Инструментарий создания информационной системы с Web-интерфейсом.
5. Система проверки бухгалтерского баланса на достоверность в процессе мониторинга и анализа финансового состояния неплатежеспособных предприятий.
6. Интеллектуальный анализ данных при управлении маркетингом в торгово-производственной фирме.
7. Интеллектуальный анализ данных при мониторинге технологического процесса в информационной системе управления производством.
8. Концепция автоматизированной интеллектуальной системы анализа фондового рынка.
9. Система анализа рисков при управлении инвестиционным портфелем и кредитами в информационной системе “Банк”.
10. Методы анализа финансовых инструментов и прогнозирования с использованием нейронных сетей.
11. Система управления инвестиционным портфелем на основе генерации и анализа гипотез с использованием Байесовского подхода.
12. Автоматизированная система документооборота банка.
13. Имитационная модель транзакционных процессов коммерческого банка.
14. Разработка интеллектуальной технологии Интернет-маркетинга.
15. Разработка модели реинжиниринга бизнес-процессов в банке.
16. Разработка мультиагентных технологий в интеллектуальной информационной системе.
17. Риск-анализ инвестиционного портфеля на основе нечеткой логики.
18. Сценарный анализ рисков при управлении предприятием на базе имитационного моделирования процессов.
19. Задачи, решаемые при внутрифирменном планировании. Календарное планирование производства. Структура автоматизированной интеллектуальной системы планирования.
20. Оперативно-диспетчерское управление предприятием. Фреймово-продукционная модель представления знаний в АИС диспетчерского управления.
21. Автоматизированная интеллектуальная система управления производственными процессами.
22. Прогнозирование и планирование доходов и затрат в интеллектуальной системе маркетинга.
23. Стратегия и программа маркетинга. Маркетинговый контроль. Определение информационных потоков и структуры системы управления маркетингом.
24. Ценные бумаги и их обращение. Финансовые и предпринимательские риски.
25. Виды и критерии риска. Риск-менеджмент.
26. Механика рынка ценных бумаг. Анализ и прогнозирование рынка.
27. Технический и фундаментальный анализ в интеллектуальной информационной системе рынка ценных бумаг.
28. Модель прогнозирования развития ситуаций в интеллектуальной информационной системе рынка ценных бумаг.
29. Анализ банковской деятельности. Основы банковского кредитования. Анализ существующих банковских технологий.

30. Обоснование рекомендаций и прогнозирование в банковской интеллектуальной советующей системе.

7 Учебно-методическое обеспечение дисциплины

7.1 Рекомендуемая литература

7.1.1 Основная литература

1. Интеллектуальные системы [Электронный ресурс]: учебное пособие/ А.М. Семенов [и др.].— Электрон. текстовые данные.— Оренбург: Оренбургский государственный университет, ЭБС АСВ, 2013.— 236 с.— Режим доступа: http://www.iprbookshop.ru/30055.— ЭБС «IPRbooks», по паролю
2. Солдатова О.П., Лезина И.В. Логическое программирование на языке Visual Prolog: учебное пособие. - Самара: СНЦ РАН, 2010. - 81 с., http://window.edu.ru/resource/698/69698
3. Интеллектуальные информационные системы и технологии : учебное пособие / Ю.Ю. Громов, О.Г. Иванова, В.В. Алексеев и др. ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». - Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2013. - 244 с. : ил. - Библиогр. в кн. - ISBN 978-5-8265-1178-7 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=277713 (25.03.2016
4. Серегин, М.Ю. Интеллектуальные информационные системы : учебное пособие / М.Ю. Серегин, М.А. Ивановский, А.В. Яковлев ; Министерство образования и науки Российской Федерации, Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования «Тамбовский государственный технический университет». - Тамбов : Издательство ФГБОУ ВПО «ТГТУ», 2012. - 205 с. : ил. - Библиогр. в кн. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=277790 (25.03.2016)
5. Балдин К.В. Информационные системы в экономике [Электронный ресурс]: учебник/ Балдин К.В., Уткин В.Б.— Электрон. текстовые данные.— М.: Дашков и К, 2015.— 395 с.— Режим доступа: http://www.iprbookshop.ru/24785.— ЭБС «IPRbooks», по паролю

ДОПОЛНИТЕЛЬНЫЕ

1. Гаскаров Д.В. Интеллектуальные информационные системы [Текст] : учеб. для вузов /. - М. : [б. и.], 2003, Высшая школа. - 431 с. (НТБ ГАГУ, 6 экз.)
2. Макаренко С.И. Интеллектуальные информационные системы: Учебное пособие. - Ставрополь: СФ МГГУ им. М.А. Шолохова, 2009. - 206 с., http://window.edu.ru/resource/462/79462
3. Дубровин А.Д. Интеллектуальные информационные системы: Учебное пособие для студентов факультета МИСИТ Московского государственного университета культуры и искусств, обучающихся по специальности 080801 - Прикладная информатика (в менеджменте) / Науч. ред. О.В. Шлыкова. - М.: МГУКИ, 2008, http://window.edu.ru/resource/724/58724
4. Былкин В.Д. Основы построения и функционирования интеллектуальных информационных систем: Учебное пособие / В.Д. Былкин, В.Н.Дубинин, Т.А. Глебова, И.И. Коновалова; Под общ. ред. проф. А.Н. Кошева. - Пенза: ПГУАС, 2007 – 207 с., http://window.edu.ru/resource/960/74960

5. Телков А.Ю. Экспертные системы: Учебное пособие. - Воронеж: ИПЦ ВГУ, 2007. - 83 с., http://window.edu.ru/resource/548/59548
6. Сотник, С.Л. Проектирование систем искусственного интеллекта : курс / С.Л. Сотник. - М. : Интернет-Университет Информационных Технологий, 2007. - 204 с. : ил., табл., схем. ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=234802 (25.03.2016)
7. Матвеев, М.Г. Модели и методы искусственного интеллекта. Применение в экономике : учебное пособие / М.Г. Матвеев, А.С. Свиридов, Н.А. Алейникова. - М. : Финансы и статистика, 2011. - 448 с. - ISBN 978-5-279-03279-2 ; То же [Электронный ресурс]. - URL: http://biblioclub.ru/index.php?page=book&id=220187 (25.03.2016).

7.1.2 Дополнительная литература

1. Бессмертный И.А. Искусственный интеллект – СПб: СПбГУ ИТМО, 2010. – 132 с.
2. Абдикеев Н.М. Проектирование интеллектуальных систем в экономике: Учебник / Под ред. Н.П. Тихомирова. - М.Ж Издательство «Экзамен», 2004.
3. Андрейчиков А.В., Андрейчикова О.Н. Интеллектуальные информационные системы.: Финансы и статистика, 2004, 424 с.
4. Интеллектуальные системы поддержки принятия решений в нештатных ситуациях с использованием информации о состоянии природной среды.: сборник / В.А.Геловани, А.А Башлыков, В.Б. Бритков. - М. : Эдиториал УРСС, 2001. - 304 с. - 1000 экз. - ISBN 5-8360-0298-3 (в пер.) 384.
5. Романов В. П. Интеллектуальные информационные системы в экономике [Текст] : учебное пособие / В. П. Романов ; ред. Н. П. Тихомиров. - М. : Экзамен, 2003. - 496 с. ; 22 см. - 5000 экз. - ISBN 5-94692-194-0 (в пер.).
6. Балдин К. В., Уткин В. Б. Информационные системы в экономике: Учебник / - 3-е изд. - М. : Дашков и К, 2006 - 395 с.
7. Гаврилова Т.А., Хорошевский В.Ф. Базы знаний интеллектуальных систем. - СПб.: Питер, 2001 - 384 с.
8. Джексон П. Введение в экспертные системы. - М., СпБ., Киев: "Вильямс", 2001.
9. Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. М.: “Нолидж”, 2000.
10. Одинцов Б.Е. Проектирование экономических информационных экспертных систем.- М.: Юнити,1996.
11. Попов Э.В., Фоминых И.Б., Кисель Е.Б., Шапот М.Д. Статические и динамические экспертные системы: Учебное пособие. М.: Финансы и статистика, 1996 – 320 с.
12. Тельнов Ю.Ф. Интеллектуальные системы обработки данных: Учебное пособие. - М.: МЭСИ, 1989. - 102 с.
13. Тельнов Ю.Ф. Проектирование баз знаний. Учебное пособие. - М.: МЭСИ, 1992. - 100 с.
14. Тельнов Ю.Ф. Интеллектуальные информационные системы в экономике. М.: СИНТЕГ, 1999. - 214 с.
15. Братко И. Язык PROLOG (Пролог): алгоритмы искусственного интеллекта. - М.: Вильямс, 2000. - 640 с.
16. Шапиро Л.Д. Методическое пособие по составлению бизнес-планов с использованием программного продукта "Project Expert" по курсу "Экспертные системы в бизнес-планировании". - Томск : Изд-во ТГУ, 2001. - 10 с.
17. Интеллектуальные информационные системы [Электронный ресурс] : УМК по специальности 080801 Прикладная информатика в экономике / сост. Е. А. Петрова, А. С. Дерешева. - Электрон. текстовые дан. - Горно-Алтайск : РИО ГАГУ, 2010. - 1 эл. опт. диск (CD-ROM) ; 12 см. - 35 экз.

18. Интеллектуальные информационные системы [Текст] : УМК ОЗО обучающихся по специальности 050202 Информатика / сост. Н. В. Фотиев. - Горно-Алтайск : РИО ГАГУ, 2010. - 36 с. : табл., ил. ; 20 см. - 25 экз.

8 Образовательные технологии

При реализации программы дисциплины «Интеллектуальные информационные системы» используются различные образовательные технологии.

8.1 Использование информационных технологий

Программное обеспечение:

- При изучении данного курса для выполнения практических и лабораторных работ необходимы персональные компьютеры, объединенные в локальную сеть, с набором средств программного обеспечения: установленный на клиентских компьютерах Prolog.

Интернет-ресурсы:

1. Лекции "Интеллектуальные информационные системы", автор к.т.н., доцент Хабаров С. П., Санкт-Петербургская лесотехническая академия, Кафедра информатики и информационных систем. [Электронный ресурс] - Режим доступа: http://www.habarov.spb.ru/new_es/index.htm
2. Объектно-событийное программирование. Экспертные системы – электронный учебник. Режим доступа: http://bourabai.ru/alg/expert.htm

8.2 Применение активных методов обучения

При реализации программы «Интеллектуальные информационные системы» дисциплины используются следующие активные методы обучения:

- ***интерактивная лекция*** – лекция, которая проводится с применением следующих активных форм обучения:

- *работа с документами* - под документами (при расширительном толковании) можно понимать: тексты законов, различные юридические документы (иски, договоры, акты, доверенности и т.п.), фотографии, письма, публичные выступления, газетные и журнальные материалы, энциклопедии и справочники, словари, карты, статистические данные и т.п. В этом методе документ превращается в материал для основательного изучения и для приобретения таких навыков и умений, как пользование различными источниками и документами, работа с информацией, навыки вдумчивого (критического) мышления, а также навыки самообразования.

- *эвристическая беседа («Сократический диалог», «метод Сократа»)* - метод получил название от восходящего к Сократу метода обучения «эвристика» (гр. – нахожу, отыскиваю, открываю). Данный метод подходит для исследования текстов, проблем, обсуждения дискуссионных вопросов, раскрытия и углубления позиции студентов, развития навыков/умения аргументации. Эвристическая беседа развивает самостоятельное и вдумчивое («критическое») мышление, способствует самопознанию, выявляет и проясняет ценностные ориентиры.

Данный метод путем тщательно подготовленных и сформулированных наводящих вопросов и примеров побуждает студентов прийти к самостоятельному правильному ответу:

1. В начале обсуждения преподаватель задает вопрос, открывающий изучаемую тему. Собственно, и сама тема может быть сформулирована в виде вопроса.

2. Особое значение в этом методе имеют «особые сократовские» вопросы, которые направлены на прояснение и углубление позиции собеседника:

• прямые уточняющие, проясняющие вопросы: Что вы имели в виду? Как вы можете по-другому сформулировать свою позицию?

• вопросы, требующие аргументации: Что дало вам основание сделать такой вывод?

Какие аргументы вы можете привести в подтверждение своей позиции?

- вопросы с использованием аналогий
- вопросы, помогающие встать на другую точку зрения
- вопросы с использованием псевдовыводов или крайних позиций
- провокационные вопросы

При использовании данного метода вопрос задается всей группе, затем после небольшой паузы для обдумывания вызывается студент, а также привлекаются другие студенты для исправления, уточнения и дополнения ответа. Данный метод позволяет вовлечь в процесс работы всю группу. Данный метод применяется при изучении темы «Анализ рынка сбыта и конкурентов»;

демонстрация наглядных пособий - наглядные пособия, видео- и аудиоматериалы делают занятия более емкими и запоминающимися. В качестве таких пособий и материалов могут быть использованы: слайды, схема, таблица, картина, рисунок, фотография, книга, газета, журнал, фильм, музыка, песня, видео- и аудиозапись и т.д.

- ***дискуссия*** - смысл данного метода состоит в обмене взглядами по конкретной проблеме. Это активный метод, позволяющий научиться отстаивать свое мнение и слушать других. Дискуссии обогащают представления студентов по теме, упорядочивают и закрепляют знания. Метод дискуссии используется на практических занятиях, где обсуждаются вопросы, когда студентам нужно защитить свои позиции, другим студентам необходимо высказать свое отношение к позиции, защищающих свой проект.

Дискуссия на семинарском (практическом) занятии требует продуманности и основательной предварительной подготовки студентов. Нужны не только хорошие знания (без них дискуссия беспредметна), но также наличие у студентов умения выражать свои мысли, четко формулировать вопросы, приводить аргументы и т. д.

9 Материально-техническое обеспечение учебного процесса

Для проведения занятий по дисциплине «Интеллектуальные информационные системы» используется аудитория для лекционных занятий, оснащенная мультимедийным проектором и экраном, компьютерный класс для проведения практических занятий, где установлены программные продукты.

10 Контроль знаний студентов

10.1 Формы текущего контроля

Текущий контроль по дисциплине «Интеллектуальные информационные системы» осуществляется в ходе учебного процесса и консультирования студентов, по результатам выполнения самостоятельных работ. Основными формами текущего контроля знаний являются: выполнение заданий на практических занятиях, защита проектов.

Вопросы для контрольной работы:

1.Роль и место информационных систем в управлении экономическими объектами.
2.Структура и состав экономической информационной системы. Внутренняя экономическая информация на объекте.
3.Проблемы повышения эффективности управления экономическими объектами и их информационного обеспечения.
4.Новая информационная технология решения задач управления в информационных системах.
5.Проблемы интерпретации. Интеллектуализация компьютера.
6.Организация работы с данными и знаниями в новой информационной технологии.
7.Развитие исследований в области искусственного интеллекта (этапы; области применения; направления исследований; проблемы и перспективы).

8.Анализ причин низкой эффективности оптимизационных моделей управления и традиционных методов обработки информации. Необходимость использования систем искусственного интеллекта.
9.Экспертные системы – основная разновидность прикладных интеллектуальных систем. Инженерия знаний. Характеристика ЭС.
10. Представление знаний в виде фреймов.
11. Семантические сети.
12. Проблема представления знаний.
13. Продукционные модели. Пример продукционной системы с консеквент-выводимой архитектурой.
14. Исчисление предикатов.
15. Дедукция и индукция.
16. Процесс стандартизации при дедуктивном выводе. Пример.
17. Применение теории нечетких множеств при формализации лингвистической неопределенности и нечетких знаний.
18. Универсальное множество, функция принадлежности нечеткого множества, базовая переменная.
19. Нечеткие отношения.
20. Нечеткая и лингвистическая переменные.
21. Операции с нечеткими множествами.
22. Лингвистические критерии и отношения предпочтения.
23. Основы нейронных сетей (архитектура, модель технического нейрона, многослойный персептрон, сеть Хопфилда, самоорганизующаяся карта Кохонена).
24. Архитектура ИИС.
25. Характерные задачи, решаемые экспертами при работе в различных предметных областях.
26. Характеристика основных функциональных модулей ИИС: база знаний (БЗ), механизм вывода, объяснение, обоснование и прогнозирование, верификация, интерфейс.
27. Разработка и этапы проектирования БЗ, представление знаний в базах данных.
28. Соотношение методов представления знаний в БД и ИИС. СУБД и СУБЗ.
29. Структура БЗ.
30. Стратегия управления и механизмы вывода. Агенда-системы.
31. Системы с «классной доской».
32. Общие методы поиска решений в пространстве состояний в продукционных системах.
33. Вывод в сети фреймов.
34. Вывод в семантической сети.
35. Дедуктивные методы поиска решений.
36. Методы поиска решений в больших пространствах состояний.
37. Поиск решений в условиях неопределенности. Вероятностная байесовская логика.
38. Приближенные рассуждения, нечеткая логика.
39. Композиционное правило вывода.
40. Пример приближенных рассуждений и композиционного правила вывода.
41. Реализация функций объяснения, обоснования и прогнозирования в ИИС.
42. Этапы проектирования ИИС и стадии существования ИИС.
43. Работа инженера знаний при разработке ИИС.
44. Инструментальные средства ИИС. Выбор инструментария.
45. Интеллектуальная система при интегрированном автоматизированном управлении экономическими объектами
46. Задачи, решаемые при внутрифирменном планировании.
47. Календарное планирование производства.
48. Структура автоматизированной интеллектуальной системы планирования.

49. Использование имитационного моделирования в интеллектуальной информационной системе для реализации функций прогнозирования.
50. Имитационное моделирование сложных производственных систем с использованием сетей Петри.
51. Оперативно-диспетчерское управление предприятием.
52. Фреймово-продукционная модель представления знаний в АИС диспетчерского управления.
53. Автоматизированная интеллектуальная система управления производственными процессами.
54. Основные факторы функционирования фирмы.
55. Прогнозирование и планирование доходов и затрат в интеллектуальной системе маркетинга.
56. Стратегия и программа маркетинга. Маркетинговый контроль.
57. Определение информационных потоков и структуры системы управления маркетингом.
58. Уровень определенности решаемых задач и среды функционирования системы управления маркетингом.
59. Ценные бумаги и их обращение. Финансовые и предпринимательские риски.
60. Виды и критерии риска. Риск-менеджмент.
61. Механика рынка ценных бумаг. Анализ и прогнозирование рынка.
62. Технический и фундаментальный анализ в интеллектуальной информационной системе рынка ценных бумаг.
63. Модель прогнозирования развития ситуаций в интеллектуальной информационной системе рынка ценных бумаг.
64. Анализ банковской деятельности. Основы банковского кредитования.
65. Анализ существующих банковских технологий.
66. Технический анализ залоговых средств (недвижимость, ценные бумаги) в интеллектуальной советующей системе “Банк”.
67. Эвристический анализ залоговых средств в интеллектуальной советующей системе “Банк”.
68. Модель реализации советующих функций в банковской информационной системе.
69. Обоснование рекомендаций и прогнозирование в банковской интеллектуальной советующей системе.

Вопросы для проверки усвоения темы «Экспертные системы»

1. Какие задачи наиболее эффективно решаются с помощью экспертных систем? Назовите основные характеристики таких задач .
2. Дайте определение экспертной системы.
3. Назовите основные компоненты структуры ЭС и раскройте их функциональную роль в системе.
4. Какие категории пользователей существуют в ЭС и чем они характерны?
5. Какими качествами должен обладать эксперт, приглашаемый для создания ЭС?
6. Какова роль и основные функции когнитолога ЭС?
7. Попытайтесь составить обобщенную схему функционирования ЭС и перечислите основные информационные связи между блоками системы.
8. В чем назначение и основные функции универсального решателя ЭС?
9. Каковы роль и функции оперативной (рабочей) памяти ЭС?
10. Какова функция модуля обработки запросов?
11. Каково содержание процедуры объяснения решения?
12. Раскройте содержание процедуры объяснения решения.
13. Какова роль и функции модуля интеллектуального редактирования?
14. Какие функциональные блоки ЭС обрабатывают информацию на внутрисистемном языке?

15. Каким в идеале должен быть входной язык пользовательского интерфейса ЭС?
16. Какая информация о пользователе ЭС должна или может содержаться в базе знаний системы?
17. В чем принципиальное отличие статической ЭС от динамической?
18. Какими средствами (аппаратными и программными) обеспечивается сопряжение блока учета динамики ПО с базой знаний ЭС?
19. Раскройте функцию блока учета динамики предметной области?
20. Какие два вида ЭС предусмотрены их обобщенной типологией и чем характерен каждый из этих видов?
21. Назовите основные признаки, по которым различаются ЭС в пределах каждого вида?
22. Чем определяются функциональные возможности любой ЭС?
23. Что такое «глубина пространства правил» базы знаний ЭС?
24. В чем отличие прямого порядка решения задачи от обратного и чем завершается решение при выборе того и (или) другом порядка?
25. Чем характеризуется мощность процедурной компоненты базы знаний ЭС?
26. Назовите основные этапы технологии проектирования и разработки ЭС?
27. Что такое «жизненный цикл» ЭС и каковы его основные фазы?
28. Каково содержание этапа идентификации процесса создания базы знаний ЭС и что является объектом идентификации в этом процессе?
29. Каково содержание этапа концептуализации при разработке информационной модели предметной области (ПрО?
30. В чем сущность атрибутивного подхода к построению модели ПрО?
31. Чем отличается структурный подход к созданию модели ПрО от атрибутивного подхода?
32. Какие типы данных используются при структурном подходе к построению модели ПрО?
33. Какими свойствами должна обладать система понятий, используемая в модели ПрО?
34. Чем завершается этап концептуализации при создании модели ПрО?
35. Приведите примеры базовых семантических отношений между понятиями ПрО?
36. Что является содержанием этапа формализации модели ПрО?
37. Раскройте сущность этапа создания опытного образца проектируемой ЭС.
38. Чем определяется «интеллектуальная мощность» экспертной системы?

Примерные темы расчетно-графических заданий

Расчетно-графические задания (РГЗ) выполняются студентами самостоятельно после прохождения определенной темы, относятся к самостоятельной работе, носят учебно-исследовательской характер. По результатам выполнения РГЗ оформляется Отчет.

1 Консультирующие экспертные системы

Тема 1. Экспертная система по определению оптимальной конфигурации ПК.
Входные данные: 1) цели использования ПК; 2) пределы стоимости выбранной конфигурации; 3) фирма - изготовитель комплектующих.
Тема 2. Экспертная система по выбору программного обеспечения для ПК пользователя.
Входные данные: 1) цели использования ПК; 2) доступные ресурсы ПК; 3) пределы стоимости требуемых приложений.
Тема 3. Экспертная система по выбору тарифного плана данного оператора сотовой связи (например, МТС). Входные данные: 1) какие вызовы преобладают: - входящие; - исходящие; - на телефоны ГТС; 2) наличие абонентской платы; 3) размер абонентской платы; 4) бесплатный определитель.
Тема 4. Экспертная система по выбору оператора сотовой связи. Входные данные: 1) зона уверенного приема сигнала; 2) стоимость роуминга; 3) предоставляемые услуги SMS, MMS, WAP 4) тарифные планы.

Тема 5. Экспертная система по выбору оборудования для компьютерной сети. Входные данные: 1) количество компьютеров в сети; 2) топология сети; 3) пределы стоимости требуемого оборудования.
Тема 6. Экспертная система по выбору программного обеспечения для предприятия. Входные данные: 1) назначение: бухгалтерия; склад; производство; продажи; … 2) масштаб предприятия: количество подразделений, филиалов и т.п.; 3) пределы стоимости ПО.
Тема 7. Экспертная система по выбору пакета прикладных программ для решения задачи на ЭВМ. Входные данные: 1) предметная область: экономика; статистика; физика; 2) математическая модель (если есть): нелинейные уравнения; обыкновенные дифференциальные уравнения; дифференциальные уравнения в частных производных; 3) требуемый результат: таблицы значений; функциональные зависимости; графики; …
Тема 8. Экспертная система по выбору программного обеспечения для компьютерной сети. Входные данные: 1) ресурсы оборудования (размеры памяти, быстродействие и т.п.); 2) требования разграничения доступа в сети; 3) фирма-производитель.
Тема 9. Экспертная система по выбору литературы (для написания реферата, курсовой работы и т.п.) в библиотеке. Входные данные: 1) предметная область; 2) ключевые слова; 3) вид литературы: справочная; учебная; художественная; статьи из журналов; методические указания.
Тема 10. Экспертная система по выбору специальности для обучения в ВУЗе. Входные данные: 1) предыдущее образование: профильный класс (физико-математический, естественно-научный и т.п.); колледж (полученная специальность); 2) в какой сфере деятельности хотели бы работать после окончания ВУЗа 3) вид специальности, которую хотели бы получить: инженерная; физико-математическая; экономическая; 4) результаты профориентационных тестов.

2 Диагностические экспертные системы

Тема 11. Экспертная система анализа текущей успеваемости студентов на факультете. Входные данные: 1) количество успевающих студентов; 2) количество студентов, имеющих более 50 % неудовлетворительных оценок и незачетов; 3) данные студентов, имеющих более 50 % неудовлетворительных оценок и незачетов. Выходные данные: 1) группы и дисциплины, по которым имеются низкие показатели успеваемости; 2) возможные причины низкой успеваемости (например, «больше 50 % студентов не прошли промежуточный контроль).
Тема 12. Экспертная система для анализа итогов сессии на факультете. Входные данные: результаты итогов рейтингового контроля и сессии по группам. Выходные данные: возможные причины низкой успеваемости студентов (например «90 % студентов, не сдавших экзамен по высшей математике, по итогам рейтингового контроля № 2, имели по данному предмету оценки «3 и ниже»).
Тема 13. Экспертная система для оценки перспективы выхода на сессию для данного студента. Входные данные: 1) результаты текущего контроля: рейтинги; контрольные работы; лабораторные работы; курсовые работы и т.п. 2) количество пропущенных занятий по каждому предмету; 3) предметы, по которым есть задолженности; 4) общее количество предметов; 5) количество дней до сессии. Выходные данные: 1) вероятность того, что студент будет допущен к сессии; 2) рекомендации, как улучшить положение дел, например, «отработать 4 часа занятий по физкультуре, сдать лабораторные работы № 5, 7 по физике и № 4 по численным методам»
Тема 14. Экспертная система для оценки степени подготовленности студента к экзамену. Входные данные: 1) количество сданных лабораторных работ, в процентах; 2) на сколько процентов выполнена курсовая работа, реферат, РГЗ по данному предмету; 3) тест по теоретическому курсу. Выходные данные: 1) вероятность того, что студент будет допущен

к сессии; 2) рекомендации по подготовке к экзамену, например, «сдать отчет по РГЗ, выучить тему «Нейронные сети».
Тема 15. Экспертная система для диагностики состояния здоровья студентов (на факультете, в ВУЗе). Входные данные: 1) количество пропусков по болезни; 2) наличие заболеваний определенного вида: дыхательных путей; опорно-двигательного аппарата; нарушения зрения; … 3) длительность пропусков по болезни. Выходные данные: 1) возможные причины возникновения наиболее распространенных заболеваний среди студентов; 2) рекомендации по улучшению состояния здоровья студентов.
Тема 16. Экспертная система для анализа обеспеченности литературой учебного процесса Входные данные: 1) наличие литературы в библиотеке; 2) потребности факультетов в учебной литературе; 3) финансовое обеспечение процесса закупки литературы; Выходные данные: 1) процент обеспеченности литературой для каждого факультета; 2) данные о том, литературы каких видов (справочники, учебники, методические пособия) и по каким предметам недостаточно для обеспечения учебного процесса.
Тема 17. Экспертная система для диагностики неисправности ЭВМ. Входные данные: 1) какие программы «неправильно» работают; 2) как изменилось быстродействие системы; 3) работа каких устройств нарушена. Выходные данные: 1) возможная причина неисправности; 2) возможные способы устранения неисправности.
Тема 18. Экспертная система для анализа потребностей предприятия в сотрудниках. Входные данные: 1) количество специалистов в той или иной области, имеющихся на предприятии; 2) работы, выполняемые предприятием; 3) трудоемкость работ. Выходные данные: 1) специалистов в какой области недостаточно; 2) какие специалисты не востребованы в данный момент; 3) рекомендации по оптимальному распределению трудовых ресурсов.
Тема 19. Экспертная система для диагностики загруженности ресурсов ЭВМ. Входные данные: 1) распределение процессорного времени по задачам; 2) приоритеты задач; 3) фрагментация жесткого диска; 4) свободное место на жестком диске; … Выходные данные: 1) состояние ресурсов системы; 2) рекомендации по оптимизации функционирования ЭВМ.
Тема 20. Экспертная система для диагностики технологического процесса. Входные данные: 1) элементы технологической цепочки; 2) отклонения выпускаемых изделий от нормы; 3) на каком этапе производства обнаружены отклонения; 4) технологические режимы. Выходные данные: 1) возможные элементы технологической цепочки, в которых произошел сбой; 2) способы устранения неисправности.

10.2 Формы промежуточного контроля

Аттестация по дисциплине проводится в форме «Зачет с оценкой».

Вопросы к зачету

Перечень вопросов к экзамену (зачету)	ПК-2
1. Роль и место информационных систем в управлении экономическими объектами.	
2. Структура и состав экономической информационной системы. Внутренняя экономическая информация на объекте.	+
3. Проблемы повышения эффективности управления экономическими объектами и их информационного обеспечения.	+
4. Новая информационная технология решения задач управления в информационных системах.	+

5. Проблемы интерпретации. Интеллектуализация компьютера.	+
6. Организация работы с данными и знаниями в новой информационной технологии.	+
7. Развитие исследований в области искусственного интеллекта (этапы; области применения; направления исследований; проблемы и перспективы).	+
8. Анализ причин низкой эффективности оптимизационных моделей управления и традиционных методов обработки информации. Необходимость использования систем искусственного интеллекта.	+
9. Экспертные системы – основная разновидность прикладных интеллектуальных систем. Инженерия знаний. Характеристика ЭС.	+
10. Проблема представления знаний.	+
11. Представление знаний в виде фреймов.	+
12. Семантические сети.	+
13. Продукционные модели. Пример продукционной системы с консеквент-выводимой архитектурой.	+
14. Исчисление предикатов.	+
15. Дедукция и индукция.	+
16. Процесс стандартизации при дедуктивном выводе. Пример.	+
17. Применение теории нечетких множеств при формализации лингвистической неопределенности и нечетких знаний.	+
18. Универсальное множество, функция принадлежности нечеткого множества, базовая переменная.	+
19. Нечеткие отношения.	+
20. Нечеткая и лингвистическая переменные.	+
21. Операции с нечеткими множествами.	+
22. Лингвистические критерии и отношения предпочтения.	+
23. Основы нейронных сетей (архитектура, модель технического нейрона, многослойный персептрон, сеть Хопфилда, самоорганизующаяся карта Кохонена).	+
24. Архитектура ИИС.	+
25. Характерные задачи, решаемые экспертами при работе в различных предметных областях.	+
26. Характеристика основных функциональных модулей ИИС: база знаний (БЗ), механизм вывода, объяснение, обоснование и прогнозирование, верификация, интерфейс.	+
27. Разработка и этапы проектирования БЗ, представление знаний в базах данных.	+
28. Соотношение методов представления знаний в БД и ИИС. СУБД и СУБЗ.	+
29. Структура БЗ.	+

30. Стратегия управления и механизмы вывода. Агенда-системы.	+
31. Общие методы поиска решений в пространстве состояний в продукционных системах.	+
32. Вывод в сети фреймов.	+
33. Вывод в семантической сети.	+
34. Дедуктивные методы поиска решений.	+
35. Методы поиска решений в больших пространствах состояний.	+
36. Поиск решений в условиях неопределенности. Вероятностная байесовская логика.	+
37. Приближенные рассуждения, нечеткая логика.	+
38. Композиционное правило вывода.	+
39. Пример приближенных рассуждений и композиционного правила вывода.	+
40. Реализация функций объяснения, обоснования и прогнозирования в ИИС.	+
41. Этапы проектирования ИИС и стадии существования ИИС.	+
42. Работа инженера знаний при разработке ИИС.	+
43. Инструментальные средства ИИС. Выбор инструментария.	+
44. Интеллектуальная система при интегрированном автоматизированном управлении экономическими объектами	+

10.3 Модульно-рейтинговая система оценки знаний студентов

Оценка знаний студентов производиться на основе модульно-рейтинговой системы. На рисунке 1 представлены блоки, из которых слагается общая оценка. Учебно-организационный блок отражает посещаемость студентами лекционных и практических занятий. Практическо - исследовательский отражает активность на практических занятиях и результаты выполнения части плана самостоятельной работы. Теоретический блок предполагает проверку знаний в форме коллоквиумов, тестов и других видов работ во время первой и второй промежуточных аттестаций.

Баллы по каждому блоку, которые может набрать студент в период изучения дисциплины, при форме контроля – «зачет с оценкой» представлены в таблице.

Таблица

Блок	Первая промежуточная аттестация	Вторая промежуточная аттестация	Итоговая промежут. аттестация	Всего
Теоретический	8	8	20	36
Практическо-исследовательский	14	14	20	48
Учебно-организационный	8	8		16
ИТОГО	30	30	40	100

Студентам необходимо помнить:

- Зачет выставляется на основе результатов первой, второй промежуточных аттестаций;
- Более 60 баллов – оценка «удовлетворительно»;

- Более 75 баллов – оценка «хорошо»;
- Более 85 баллов – оценка «отлично»
- Первая аттестация проводиться в середине изучения курса, вторая в конце курса;
- После первой аттестации необходимо обратить внимание на возможность активизации работы на практических занятиях, с целью получения максимально возможного количества баллов;
- После второй аттестации преподаватель не принимает письменные работы студентов (доклады, рефераты).

Примерные баллы по различным видам работ, выставляемые преподавателем:

- Доклад, презентация, тест - 5-10 баллов.
- Выступление на семинаре 1-2 балла.
- Выполнение лабораторной работы с отчетом 1-2 балла.
- Участие в конференции 7 баллов.
- Публикация статьи: 8 баллов.
- Участие в олимпиаде: 10 баллов

Первая промежуточная аттестация будет проводиться после прослушивания студентами лекционного материала и проведения практических занятий по первому модулю. Вторая промежуточная аттестация будет в конце второго модуля.

Составитель: Юхтина Т.И., к.ф.-м.н., доцент

И.о. зав. кафедрой экономики предприятия и прикладной информатики _____Е.А. Петрова, к.э.н., доцент