МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«ВОРОНЕЖСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
БОРИСОГЛЕБСКИЙ ФИЛИАЛ
(БФ ФГБОУ ВО «ВГУ»)

УТВЕРЖДАЮ

Заведующий кафедрой
естественнонаучных и
общеобразовательных дисциплин

С.Е. Зюзин

01.09.2018 г.

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ
Б1.В.20 Основы искусственного интеллекта

1. Код и наименование направления подготовки:

44.03.05 Педагогическое образование (с двумя профилями подготовки)

2. Профили подготовки:

Математика. Информатика и информационные технологии в образовании

3. Квалификация выпускника: бакалавр

4. Форма обучения: очная, заочная

5. Кафедра, отвечающая за реализацию дисциплины: естественнонаучных и общеобразовательных дисциплин

6. Составитель программы: Л.В. Лободина, кандидат педагогических наук, доцент

7. Рекомендована: научно-методическим советом Филиала (протокол № 1 от 31.08.2018 г.)

8. Учебный год: 2021-2022 **Семестр:** 10

9.Цели и задачи учебной дисциплины:
Целью изучения дисциплины «Основы искусственного интеллекта» является овладение систематизированными знаниями об основных моделях, методах, средствах и языках, используемых при разработке систем искусственного интеллекта.
Задачи учебной дисциплины:
сформировать умения ориентироваться в различных типах интеллектуальных систем, в различных методах представления знаний, переходить от одного метода представления знаний к другому;
– сформировать умения ставить задачу построения экспертной системы для решения задачи выбора вариантов в плохо формализуемой предметной области;
– сформировать навыки логического программирования на языке Пролог.
При проведении учебных занятий по дисциплине обеспечивается развитие у обучающихся навыков командной работы, межличностной коммуникации, принятия решений.

10. Место учебной дисциплины в структуре образовательной программы:
Дисциплина «Основы искусственного интеллекта» входит в блок Б1 «Дисциплины (модули)» и является обязательной дисциплиной вариативной части образовательной программы. Для освоения дисциплины «Основы искусственного интеллекта» необходимы знания, умения, навыки, сформированные в ходе изучения дисциплин «Математическая логика и теория алгоритмов», «Алгебра и теория чисел», «Компьютерные сети, Интернет и мультимедиа технологии», «Элементы абстрактной и компьютерной алгебры». Дисциплина «Основы искусственного интеллекта» входит в число дисциплин окончательного формирования профессиональных компетенций выпускника и готовит студентов к дальнейшей профессиональной деятельности.
Условия реализации дисциплины для лиц с ОВЗ определяются особенностями восприятия учебной информации и с учетом индивидуальных психофизических особенностей.

11. Планируемые результаты обучения по дисциплине/модулю (знания, умения, навыки), соотнесенные с планируемыми результатами освоения образовательной программы (компетенциями выпускников):

Компетенция		Планируемые результаты обучения
Код	Название	
ПК-3	способность решать задачи воспитания и духовно-нравственного развития обучающихся в учебной и внеучебной деятельности	**знает:** – задачи воспитания и духовно-нравственного развития обучающихся в учебной и внеучебной деятельности на соответствующих ступенях общего образования; **умеет:** – применять теоретические знания дисциплины (*основные понятия искусственного интеллекта; модели представления знаний; основные приемы и принципы логического программирования; синтаксис языка Prolog; принципы разработки и создания экспертных систем и экспертных оболочек; навыки логического проектирования баз знаний предметной области; навыки логического программирования на языке Visual Prolog*) для решения практических задач воспитания и духовно-нравственного развития обучающихся в учебной и внеучебной деятельности на соответствующих ступенях общего образования; **владеет:** – навыками постановки цели, формулировки задач и прогнозирования духовно-нравственного развития и воспитания личности обучающегося (воспитанника);

12. Объем дисциплины в зачетных единицах/час. — 2/72.

Форма промежуточной аттестации зачёт.

13. Виды учебной работы

Очная форма обучения

Вид учебной работы	Трудоемкость (часы)	
	Всего	По семестрам
		10 сем.
Контактная работа, в том числе:	38	38
лекции	12	12
лабораторные работы	26	26
Самостоятельная работа	34	34
Форма промежуточной аттестации *(зачёт – 0 час.)*	0	0
Итого:	72	72

Заочная форма обучения

Вид учебной работы	Трудоемкость (часы)	
	Всего	По семестрам
		10 сем.
Контактная работа, в том числе:	10	10
лекции	4	4
лабораторные работы	6	6
Самостоятельная работа	58	58
Форма промежуточной аттестации: *(зачёт – 4 час.)*	4	4
Итого:	72	72

13.1. Содержание дисциплины

№ п/п	Наименование раздела дисциплины	Содержание раздела дисциплины
1. Лекции		
1.1	Базовые понятия ИИ. Основные направления исследования в области искусственного интеллекта.	Терминология: понятия интеллекта, системы знаний, интеллектуальных задач. Философские аспекты проблемы систем ИИ. История развития систем ИИ: основные направления и разработки в области искусственного интеллекта, современные разработки систем искусственного интеллекта.
1.2	Системы знаний. Модели представления знаний.	Интеллектуальные системы. Функции и структура современных интеллектуальных систем. Отличие знаний от данных. Модели представления знаний: логическая модель; сетевая модель; семантические сети; фреймовая модель; продукционная модель представления знаний. Достоинства и недостатки разных моделей
1.3	Понятие об экспертной системе	Общая характеристика экспертных систем (ЭС). Структура и режимы использования ЭС. Классификация инструментальных средств в ЭС. Организация знаний в ЭС. Виды экспертных систем. Типы задач, решаемые в экспертных системах. Интеллектуальные информационные экспертные системы. Примеры современных ЭС, области их применения.
1.4	Общие сведения о логическом программировании.	Отличие процедурных и декларативных языков программирования. Представление знаний о предметной области в виде фактов и правил базы знаний Пролога. Дескриптивный, процедурный и машинный смысл программы на Прологе. Алгоритм выполнения программ на Прологе.

		Предикат отсечения и управление поиском с возвратом в программах на Прологе. Рекурсия и структуры данных в программах на Прологе. Обработка списков в Прологе. Решение логических задач в Прологе. Работа с бинарными деревьями в Прологе.
3. Лабораторные работы		
2.4	Лабораторная работа №1	Создание простейших проектов в среде Visual Prolog. Среда Visual Prolog: основные понятия, интерфейс. Пункт меню File. Пункт меню Edit. Пункт меню Project. Команды меню Options. Структура программы на ПРОЛОГе.
	Лабораторная работа №2	Поиск с возвратом. Работа с базами данных на ПРОЛОГе. Управление поиском с возвратом: предикаты fail и отсечения.
	Лабораторная работа №3	Арифметические вычисления. Встроенные функции для вычисления арифметических выражений на ПРОЛОГе. Предикат для ввода действительных чисел.
	Лабораторная работа №4	Рекурсия. Организация циклов: механизм отката и рекурсия. Точки возврата. Предки и потомки. Доказательство рекурсивной цели.
	Лабораторная работа №5	Решение логических задач в Prolog. Логические задачи как конечные множества с одинаковым количеством элементов, между которыми устанавливается взаимно-однозначное соответствие. Описание конечных множеств как баз данных. Установление зависимости между объектами с помощью правил.
	Лабораторная работа №6	Списки. Составные объекты в Visual Prolog: списки и деревья. Хвост и голова списка. Корни и листья деревьев. Узлы.

13.2. Темы (разделы) дисциплины и виды занятий

Очная форма обучения

№ п/п	Наименование темы (раздела) дисциплины	Виды занятий (часов)				
		Лекции	Практические	Лабораторные	Самостоятельная работа	Всего
1.	Базовые понятия ИИ. Основные направления исследования в области искусственного интеллекта.	2	0	0	4	6
2.	Системы знаний. Модели представления знаний.	2	0	0	6	8
3.	Понятие об экспертной системе	2	0	0	6	8
4.	Общие сведения о логическом программировании.	6	0	0	6	12
4.1	Лабораторная работа №1	0	0	4	2	6
4.2	Лабораторная работа №2	0	0	4	2	6
4.3	Лабораторная работа №3	0	0	4	2	6
4.4	Лабораторная работа №4	0	0	4	2	6
4.5	Лабораторная работа №5	0	0	6	2	8
4.6	Лабораторная работа №6	0	0	4	2	6
	Зачёт					0
	Итого:	12	0	26	34	72

Заочная форма обучения

№ п/п	Наименование темы (раздела) дисциплины	Виды занятий (часов)				
		Лекции	Практические	Лабораторные	Самостоятельная работа	Всего
1.	Базовые понятия ИИ. Основные направления исследования в области искусственного интеллекта.	1	0	0	4	5
2.	Системы знаний. Модели представления знаний.	1	0	0	6	7
3.	Понятие об экспертной системе	1	0	0	6	7
4.	Общие сведения о логическом программировании.	1	0	0	6	7
4.1	Лабораторная работа №1	0	0	1	6	7
4.2	Лабораторная работа №2	0	0	1	6	7
4.3	Лабораторная работа №3	0	0	1	6	7
4.4	Лабораторная работа №4	0	0	1	6	7
4.5	Лабораторная работа №5	0	0	1	6	7
4.6	Лабораторная работа №6	0	0	1	6	7
	Зачёт					4
	Итого:	4	0	6	58	72

14. Методические указания для обучающихся по освоению дисциплины

Приступая к изучению учебной дисциплины, целесообразно ознакомиться с учебной программой дисциплины, электронный вариант которой размещён на сайте БФ ВГУ.

Знание основных положений, отраженных в рабочей программе дисциплины, поможет обучающимся ориентироваться в изучаемом курсе, осознавать место и роль изучаемой дисциплины в подготовке будущего выпускника, строить свою работу в соответствии с требованиями, заложенными в программе.

Основными формами контактной работы по дисциплине являются лекции и лабораторные работы, посещение которых обязательно для всех студентов (кроме студентов, обучающихся по индивидуальному плану).

Подготовка к лабораторным работам ведется на основе планов лабораторных работ, которые размещены на сайте филиала. В ходе подготовки к лабораторным работам необходимо изучить в соответствии с вопросами для повторения основную литературу, ознакомиться с дополнительной литературой. Кроме того, следует повторить материал лекций, ответить на контрольные вопросы, изучить образцы решения задач, выполнить упражнения (если такие предусмотрены).

При подготовке к промежуточной аттестации необходимо повторить пройденный материал в соответствии с учебной программой, примерным перечнем вопросов, выносящихся на зачет. Рекомендуется использовать конспекты лекций и источники, перечисленные в списке литературы в рабочей программе дисциплины, а также ресурсы электронно-библиотечных систем.

Для достижения планируемых результатов обучения используются интерактивные лекции, анализ имитационных моделей.

15. Перечень основной и дополнительной литературы, ресурсов интернет, необходимых для освоения дисциплины

а) основная литература:

№ п/п	Источник
1	Боровская Е.В. и др. Основы искусственного интеллекта: учеб. пос.- М.: Бином, 2010
2	Сотник, С.Л. Проектирование систем искусственного интеллекта : курс / С.Л. Сотник. - Москва : Интернет-Университет Информационных Технологий, 2007. - 204 с. : ил., табл., схем. ; То же [Электронный ресурс]. –

	URL: http://biblioclub.ru/index.php?page=book&id=234802 (11.01.2018).

б) дополнительная литература:

№ п/п	Источник
3	Смолин Д.В. Введение в искусственный интеллект: конспект лекций.- 2-е изд., перераб.- М.: ФИЗМАТЛИТ, 2007
4	Ясницкий Л.Н. Введение в искусственный интеллект: учеб. пос. для вузов.- М.: Академия, 2005

в) информационные электронно-образовательные ресурсы**:**

№ п/п	Источник
5	Бессмертный И.А. Искусственный интеллект: Учебное пособие. - СПб: СПбГУ ИТМО, 2010. - 132 с. http://window.edu.ru/resource/274/69274/files/itmo443.pdf (11.01.2018).
6	Туганбаев, А.А. Линейная алгебра : учебное пособие / А.А. Туганбаев. - Москва : Флинта, 2012. - 74 с. - ISBN 978-5-9765-1407-2 ; То же [Электронный ресурс]. – URL: http://biblioclub.ru/index.php?page=book&id=115141 (11.01.2018).

16. Перечень учебно-методического обеспечения для самостоятельной работы

№ п/п	Источник
1	Методические материалы по дисциплине «Основы искусственного интеллекта» - http://bsk.vsu.ru/sveden/education#OOP
2	Системы искусственного интеллекта: Методические указания к выполнению лабораторных работ / Сост. Гудков П.А. - Пенза: Пензенский гос. ун-т, 2007. - 53 с. http://window.edu.ru/resource/709/59709/files/stup420.pdf (11.01.2018).

17. Информационные технологии, используемые для реализации учебной дисциплины, включая программное обеспечение, информационно-справочные системы и профессиональные базы данных

программное обеспечение:

–Win10 (или Win7), OfficeProPlus 2010
–браузеры: Yandex, Google, Opera, Mozilla Firefox, Explorer
–STDU Viewer version 1.6.2.0
–7-Zip
–GIMP GNU Image Manipulation Program
–Paint.NET
–Tux Paint
–Adobe Flash Player
–Visual Prolog

информационно-справочные системы и профессиональные базы данных:

–Научная электронная библиотека – http://www.scholar.ru/;
–Федеральный портал Российское образование – http://www.edu.ru/;
–Информационная система «Единое окно доступа к образовательным ресурсам» http://window.edu.ru/;
–Федеральный центр информационно-образовательных ресурсов – http://fcior.edu.ru;
–Лекции ведущих преподавателей вузов России в свободном доступе – https://www.lektorium.tv/;
–Электронно-библиотечная система «Университетская библиотека online» – http://biblioclub.ru/.

18. Материально-техническое обеспечение дисциплины:

Мультимедийное оборудование (проектор, ноутбук или стационарный компьютер, экран), компьютерный класс (компьютеры, объединенные в сеть с выходом в Интернет и обеспечением доступа в электронную информационно-образовательную среду ВГУ и БФ).

19. Фонд оценочных средств:

19.1. Перечень компетенций с указанием этапов формирования и планируемых результатов обучения

Код и содержание компетенции (или ее части)	Планируемые результаты обучения (показатели достижения заданного уровня освоения компетенции посредством формирования знаний, умений, навыков)	Этапы формирования компетенции (разделы (темы) дисциплины или модуля и их наименование)	ФОС* (средства оценивания)
ПК-3: способность решать задачи воспитания и духовно-нравственного развития обучающихся в учебной и внеучебной деятельности	Знать: – задачи воспитания и духовно-нравственного развития обучающихся в учебной и внеучебной деятельности на соответствующих ступенях общего образования.	1. Базовые понятия ИИ. Основные направления исследования в области ИИ. 2. Системы знаний. Модели представления знаний. 3. Понятие об экспертной системе	Контрольная работа Тест
	Уметь: – применять теоретические знания дисциплины (*основные понятия искусственного интеллекта; модели представления знаний; основные приемы и принципы логического программирования; синтаксис языка Prolog; принципы разработки и создания экспертных систем и экспертных оболочек; навыки логического проектирования баз знаний предметной области; навыки логического программирования на языке Visual Prolog*) для решения практических задач воспитания и духовно-нравственного развития обучающихся в учебной и внеучебной деятельности на соответствующих ступенях общего образования.	1. Базовые понятия ИИ. Основные направления исследования в области ИИ. 2. Системы знаний. Модели представления знаний. 3. Понятие об экспертной системе	Тест Комплекты индивидуальных заданий для практических работ Доклады, сообщения
	Владеть: – навыками постановки цели, формулировки задач и прогнозирования духовно-нравственного развития и воспитания личности обучающегося (воспитанника).	1. Базовые понятия ИИ. Основные направления исследования в области ИИ. 2. Системы знаний. Модели представления знаний. 3. Понятие об экспертной системе	Комплекты индивидуальных заданий для практических работ Разноуровневые задания Тесты
Промежуточная аттестация – зачёт			Комплект КИМ

19.2 Описание критериев и шкалы оценивания компетенций (результатов обучения) при промежуточной аттестации

Для оценивания результатов обучения на зачете используются следующие показатели (ЗУНы из 19.1):

1) знание основ дисциплины (*основные понятия искусственного интеллекта; модели представления знаний; основные приемы и принципы логического программирования; синтаксис языка Prolog; принципы разработки и создания экспертных систем и экспертных оболочек*);

2) умение связывать теорию с практикой;
3) навыки логического программирования на языке Visual Prolog.
Для оценивания результатов обучения на зачете используется – зачтено, не зачтено

Соотношение показателей, критериев и шкалы оценивания результатов обучения.

Критерии оценивания компетенций	Уровень сформированности компетенций	Шкала оценок
Обучающийся в полной мере владеет теоретическими основами дисциплины, способен иллюстрировать ответ примерами, фактами, данными научных исследований, применять теоретические знания для логического программирования на языке Visual Prolog типовых задач и практических заданий более высокого уровня сложности *ИЛИ* *Обучающийся владеет теоретическими основами дисциплины, способен иллюстрировать ответ примерами, фактами, применять теоретические знания для логического программирования на языке Visual Prolog типовых задач* *ИЛИ* *Обучающийся частично владеет теоретическими основами дисциплины, иногда затрудняется иллюстрировать ответ примерами, фактами, не всегда способен применять теоретические знания для логического программирования на языке Visual Prolog типовых задач*	*Повышенный уровень* *Базовый уровень* *Пороговый уровень*	*Зачтено*
Обучающийся демонстрирует отрывочные, фрагментарные знания, допускает грубые ошибки при решении типовых задач либо не имеет представления о способе их решения.	–	*Не зачтено*

19.3 Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности, характеризующие этапы формирования компетенций в процессе освоения образовательной программы

19.3.1 Перечень вопрос к зачёту:

1. Базовые понятия искусственного интеллекта.
2. Работа со строками в Visual Prolog (некоторые стандартные предикаты).
3. История развития ИИ и основные направления исследований в области ИИ
4. Составные объекты в Visual Prolog: деревья.
5. Философские аспекты проблемы ИИ (возможность существования, безопасность, полезность).
6. Составные объекты в Visual Prolog: списки.
7. Интеллектуальные системы. Отличия знаний от данных. Свойства знаний.
8. Организация циклов: рекурсия.
9. Структура и функции интеллектуальных систем. Разновидности интеллектуальных систем.
10. Управление поиском с возвратом (предикаты ! и fail).
11. Модели представления знаний. Семантические сети. Достоинства и недостатки семантических сетей.
12. Поиск с возвратом в Visual Prolog.
13. Представление знаний на основе фреймов. Структура фреймов. Системы фреймов. Достоинства и недостатки фреймового представления.
14. Структура программы на Visual Prolog и стандартные типы данных.
15. Логические модели представления знаний.
16. Предикаты и переменные в Visual Prolog.
17. Продукционные модели представления знаний. Достоинства и недостатки продукционной модели.
18. Предложения: факты и правила языка Visual Prolog. Запросы (цели) в Visual Prolog.
19. Экспертные системы. Архитектура современных экспертных систем.
20. Процедурный и декларативный смысл программы на Прологе.

21. Пример создания экспертной системы.
22. Сравнительная характеристика декларативных и процедурных языков программирования. Основные отличия, области применения.

19.3.2 Комплекты заданий для практических работ

Не предусмотрены

19.3.3 Комплект тестовых заданий

1. Язык Пролог является
+декларативным языком программирования
-процедурным языком программирования

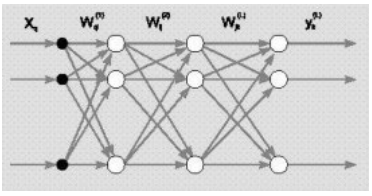
2. В Прологе правило, состоящее только из первой части называется:
+фактом
-вопросом
-предикатом

3. В Прологе вместо слова and используется
+запятая
-двоеточие, тире
-точка с запятой

4. Раздел, в который заносятся факты и правила, изначально известные, называется:
+clauses
-predicates
-domains
-goal

5. Нейроны, принимающие сигналы от одних нейронов и передающие другим нейронам, называются...
-рецепторы
+внутренние нейроны
-реагирующие нейроны

6. Сеть изображенная на рисунке является...



-однослойной
-двухслойной
+трехслойной
-четырехслойной

7. Основным компонентом экспертной системы является...
-блок приобретения знаний
-интерфейс
+база знаний
-подсистема объяснений
-сиситема логического выбора
-база данных

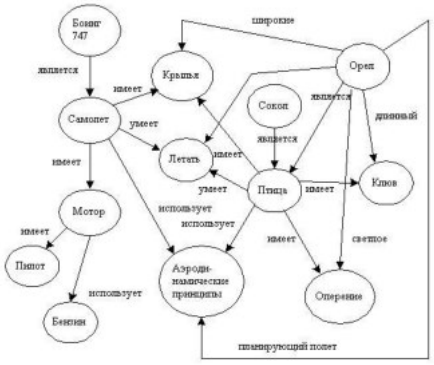
8. База интеллектуальной системы, которая содержит знания о самой себе называется...
+базой метазнаний
-базой целей
-базой процедур
-базой правил

-базой фактов
-базой закономерностей

9. Блок решателя, который отвечает за решение задач расчетно-логического и алгоритмического типа,называется
-блоком планирования
+блоком функциональных преобразований
-блоком дедективного вывода
-блоком индуктивного вывода

10. Из перечисленных компонент интеллектуальной системы выберете те, которые реализуют функции общения.
+естественно-языковой интерфейс
+рецепторы
+эффекторы
-монитор базы знаний
-монитор решателя

11. Модель представления знаний, изображенная на рисунке называется...



+семантической сетью
-продукционной моделью
-логической моделью
-фреймом

12. Интеллектом, в теории искусственного интеллекта, называется...
+способность мозга решать интеллектуальные задачи путем приобретения, запоминания и целенаправленного преобразования знаний в процессе обучения на опыте
-способность мозга решать задачи путем приобретения и целенаправленного преобразования знаний в процессе обучения на опыте
-способность мозга решать интеллектуальные задачи путем приобретения, запоминания и целенаправленного преобразования данных в процессе обучения на опыте

13. Раздел, в котором формулируется назначение создаваемой программы на языке Пролог, называется...
+predicates
-domains
-clauses
-goal

14. В Прологе данные, которые принимаются за истину и не требуют доказательства, называются...
-правилами
+фактами
-предложениями

15. Процесс выработки в некоторой системе той или иной реакции на группы внешних идентичных сигналов путем многократного воздействия называется

-самообучением
+обучением
-адаптацией

16. Предикат, который включает поиск с возвратом при его отсутствии, называется
-отсечением
+fail
-repeat

17. Секция в которой описываются нестандартные домены называется...
-Predicates
+Domains
-Clauses
-Goal

Критерии оценки результата

Каждое правильно выполненное задание – 1 б.
«5» – 15-17 баллов, «4» – 11-14 баллов,
«3» – 8-13 баллов, «2» – 0-7 баллов

19.3.4 Перечень заданий для контрольных работ

Тема «Знакомство с основами логического программирования (Prolog)»

Содержание отчета:

- На титульном листе указать номер зачетной книжки, специальность, курс, группу, ФИО;
- задание;
- нарисовать дерево родственных отношений;
- исходные тексты;
- выводы по работе.

Задание (типовое):

Используя предикаты parent (symbol, symbol), man (symbol), woman (symbol), married (symbol,symbol), записать факты, описывающие Вашу семью. Записать 8 правил вывода для любых родственных отношений в Вашей семье (например: мать, отец, сестра, брат, племянница, племянник, тетя, дядя, внучка, внук, бабушка, дедушка, двоюродная сестра, двоюродный брат и т.д.).

При отладке программы изучить и использовать возможности трассировки.

Написать программу без использования внутренней цели (без секции GOAL).

Во всех прочих программах используется внутренняя цель (в секции GOAL)

Методические указания

Как правило, программа на Прологе состоит из двух программных секций: секции предикатов PREDICATES и секции предложений CLAUSES.

В секции PREDICATES описываются собственные предикаты. Описание предикатов заключается в их перечислении с указанием доменов (типов) их аргументов. Стандартные предикаты объявлять не требуется. Формат объявления предиката:

predicate_name (argument_domain_1, argument_domain_2, …, argument_domain_n)

В секции CLAUSES помещаются предложения – факты и правила, составляющие программу. Все предложения для одного предиката должны быть сгруппированы. Каждое предложение заканчивается точкой. Формат записи предложений:

fact(object_1, object_2, …, object_n).

rule(Var_1, Var_2, …,Var_m):–subgoal_1, subgoal_2, …, subgoal_k.

Для отладки программы можно использовать возможности трассировки. Трассировка позволяет в пошаговом режиме проследить процесс нахождения решения.

Для того чтобы включить трассировку, можно первой строкой программы поместить директиву trace. Для пошагового выполнения программы в режиме трассировки используется клавиша F10.

При работе в режиме трассировки вся текущая информация появляется в окне трассировки Trace.

Сообщения в окне трассировки могут быть следующими:

- CALL – вывод имени цели и значений ее параметров;
- FAIL – вывод имени неудачно завершившейся цели;

– REDO – вывод сообщения о том, что произведен поиск с возвратом;
– RETURN – вывод имени удачно завершившейся цели и значений ее параметров (символ * показывает, что существуют другие решения).

Примеры выполнения заданий
РЕКУРСИЯ

```
PREDICATES
factorial (integer, integer)
CLAUSES
factorial (0, 1):- !.
factorial (N, Factorial_N):- M=N–1, factorial (M, Factorial_M), Factorial_N=Factorial_M*N.
GOAL
write (“Для какого числа Вы хотите найти факториал? ”), readint (Number), factorial (Number, Result), write (Number, “!=”, Result).
```

Пример хвостовой рекурсии

```
PREDICATES
counter (integer)
CLAUSES
counter (N):- N>0, !, write (“N=”, N), nl, New_N=N–1, counter (New_N).
counter (N):- write (“Отрицательное N=”, N).
GOAL
counter (1000).
```

Пример нехвостовой рекурсии

```
PREDICATES
counter (integer)
check (integer)
CLAUSES
counter (N):- check (N), write (“N=”, N), nl, New_N=N–1, counter (New_N).
check (N):- N>0, write (“Положительное ”).
check (_):- write (“Отрицательное ”).
GOAL
counter (1000).
```

Подсчет числа элементов списка или, другими словами, определение длины списка.

```
DOMAINS
intlist=integer*
PREDICATES
list_length (intlist, integer)
CLAUSES
list_length ([ ], 0):- !.
list_length (List, Length):- List=[H | T], list_length (T, Length_T), Length=Length_T+1.
GOAL
listlength ([1, 2], Length), write(“Length=”, Length).
```

Поэлементный вывод дерева, в узлах которого хранятся целые числа,

на экран

```
DOMAINS
treetype=tree (integer, treetype, treetype) ; empty
PREDICATES
printtree (treetype)
CLAUSES
printtree (empty):- !.
```

printtree (tree (Root, Left, Right)):- write (Root), write (“ ~ ”), printtree (Left), printtree (Right).
GOAL
printtree (tree (5, tree (2, tree (–1, empty, empty), tree (4, empty, empty)), tree (9, tree(7, empty, empty), empty))).

Результат работы программы обхода дерева «сверху вниз»:

5 ~ 2 ~ –1 ~ 4 ~ 9 ~ 7 ~

19.3.5 Темы курсовых работ

Не предусмотрены

19.3.6 Темы рефератов (докладов, сообщений)

1. История развития искусственного интеллекта.
2. Возможен ли искусственный интеллект.
3. Мышление и искусственный интеллект.
4. Современная наука и искусственный интеллект.
5. Философские аспекты проблемы систем ИИ (возможность существования, безопасность, полезность).
6. Искусственный интеллект – настоящее и будущее.
7. Кибернетика и сознание.
8. Проблема обучения распознавания образов.
9. Устройство нейронных сетей.
10. Применение нейронных сетей.
11. Распознавание образов. Понятие образа. Проблема обучения распознавания образов (ОРО).
12. Геометрический подход к обучению распознавания образов.
13. Структурный подход к обучению распознавания образов.
14. Обучение и самообучение. Адаптация и обучение.
15. Нейронные сети.
16. Однослойные нейронные сети.
17. Многослойные нейронные сети
18. Обучение искусственных нейронных сетей
19. Персептроны. Персептронная представляемость. Проблема функции ИСКЛЮЧАЮЩЕГО ИЛИ.
20. Линейная разделимость. Преодоление ограничения линейной разделимости.
21. Обучение персептрона. Алгоритм обучения однослойного персептрона.

19.3.7 Комплект заданий для лабораторных работ

Лабораторная работа №1
Создание простейших проектов в среде Visual Prolog

Среда Visual Prolog: основные понятия, интерфейс.

Prolog является языком, основанным на программировании логики (PROgramming in LOGic). Вместо детальных инструкций, предписывающих как решать ту или иную задачу, программист на языке Prolog уделяет основное внимание описанию задачи.

В среде Visual Prolog используется подход, получивший название «визуальное программирование», при котором внешний вид и поведение программ определяются с помощью специальных графических средств проектирования без традиционного программирования на алгоритмическом языке.

Visual Prolog автоматизирует построение сложных процедур и освобождает программиста от выполнения тривиальных операций. С помощью Visual Prolog проектирование пользовательского интерфейса и связанных- с ним окон, диалогов, меню, линии уведомлений о состояниях и т.д. производится в графической среде. С созданными объектами сразу же могут работать различные Кодовые Эксперты (Code Experts), которые используются для генерации базового и расширенного кодов на языке Prolog, необходимых для обеспечения их функционирования.

В Visual Prolog входят интерактивная среда визуальной разработки (VDE — Visual Develop Environment), которая включает текстовый и различные графические редакторы, инструментальные средства генерации кода, конструирующие управляющую логику (Experts), а также являющийся расширением языка интерфейс визуального программирования (VPI — Visual Programming Interface), Пролог-компилятор, набор различных подключаемых файлов и библиотек, редактор связей, файлы, содержащие примеры и помощь.

Visual Prolog поддерживается различными ОС, в том числе MS-DOS, всеми версиями Windows, а также некоторыми другими системами, требующими графического пользовательского интерфейса.

Запустите Visual Prolog. Для этого нажмите кнопку **Пуск**, выберите в меню пункт **Программы**, далее пункт **Visual Prolog 5.2 Personal Edition** и в появившемся подменю – пункт **Visual Prolog.**

Интерфейс Visual Prolog включает: главное меню, панель инструментов, окно проекта. Если во время последнего использования системы Visual Prolog там был открытый проект, то система автоматически вновь откроет этот проект.

На рис.1 изображен внешний вид среды Visual Prolog после запуска. В окне проекта отображаются модули открытого проекта route.prj: karta.pro, route.pro, VPITools.pro.

Рис 1. Среда разработки Visual Prolog

Левая панель кнопок в окне проекта позволяет выбирать нужный компонент проекта: модуль, окно, меню и т.д. С помощью кнопок правой панели выбранный компонент можно редактировать(кнопка Edit), удалять(кнопка Delete), а также добавлять новый(кнопка New).

Пункт меню **File** содержит команды для работы с файлами. Чтобы создавать новое окно редактирования, можно использовать команду File | New. Эта команда создаст новое окно редактора с заголовком "NONAME".

В меню **Edit** представлены команды, позволяющие редактировать текст программы. Встроенный редактор системы по интерфейсу похож на обычный текстовый редактор. Можно производить вырезку, копирование и вставку текста, операции Отмена/Восстановление, которые можно активизировать из меню Edit. Также меню Edit показывает "горячие клавиши", связанные для этих действий.

Пункт меню **Project** содержит команды для работы с проектом: создать новый, открыть, запустить и т.д. Запуск проекта на исполнение выполняется нажатием кнопки <R> на панели инструментов (или F9, или с помощью команд меню Project | Run).

Команды меню **Options** позволяют выполнять настройку проекта, устанавливать необходимые параметры.

Программа на ПРОЛОГе состоит из предложений, которые могут быть фактами, правилами или запросами. Как правило, программа состоит из четырех разделов.

DOMAINS – секция описания доменов (типов). Секция применяется, если в программе используются нестандартные домены.

PREDICATES – секция описания предикатов. Секция применяется, если в программе используются нестандартные предикаты.

CLAUSES – секция предложений. Именно в этой секции записываются предложения: факты и правила вывода.

GOAL – секция цели. В этой секции записывается запрос.

Среда Visual Prolog позволяет протестировать программу без создания проекта. Для этого используется утилита Test Goal. Достаточно создать новый файл, набрать текст программы и активизировать Test Goal нажатием кнопки <G> на панели инструментов. Автономно исполняемый файл при этом не создается. Утилита Test Goal компилирует только тот код, который определен в активном окне редактора (код в других открытых окнах или модулях

проектов, если они есть, игнорируются). Test Goal находит *все* возможные решения задачи и автоматически выводит значения *всех* переменных.

Пример 1.

Имеется база данных, содержащая следующие факты:

родитель(Илья, Марина).
родитель(Марина, Ира).
родитель(Елена, Иван).
родитель(Николай, Ира).
родитель(Ольга, Алексей).
родитель(Марина, Саша).
родитель(Сергей, Иван).

Определить:

1) верно ли, что Марина является родителем Саши;
2) верно ли, что Алексей является родителем Ольги;
3) кто является ребенком Николая;
4) кто родители Ивана;
5) всех родителей и их детей.

Решение.

1. Запустите среду Visual Prolog. Закройте окно проекта (если оно открыто) и откройте новый файл (**File**|**New**) (рис.2)

В появившемся окне наберите текст программы, содержащий разделы: PREDICATES (описание предиката ***родитель***), CLAUSES (перечисляются имеющиеся факты) и GOAL (запрос).

Рис.2. Рабочее окно редактора

```
DOMAINS
имя=string
PREDICATES
nondeterm      родитель(имя, имя)
CLAUSES
родитель(илья, марина).
родитель(марина, ира).
родитель(елена, иван).
родитель(николай, ира).
родитель(ольга, алексей).
родитель(марина, саша).
родитель(сергей, иван).
GOAL
родитель(марина, саша) .
```

Запустите и протестируйте программу с помощью команды **Project** | **Test Goal** (можно использовать кнопку на панели инструментов **<G>** или сочетание клавиш **<Ctrl>+<G>**). Результат выполнения программы будет выведен в отдельном окне

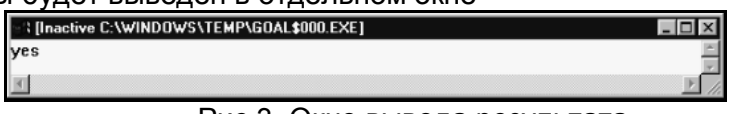


Рис.3. Окно вывода результата

Указание: перед следующим запуском программы следует закрыть это окно.

2. Для ответа на вопрос: верно ли, что Алексей является родителем Ольги, измените запрос:

GOAL
родитель(алексей, ольга).
После запуска программы (Project | Test Goal) будет получен ответ:
no
3. Для ответа на вопрос: кто является ребенком Николая, запишите цель:
GOAL
родитель(николай, X) .

Результат:
X=ира
1 Solution

4. Для ответа на вопрос: кто родители Ивана, укажите запрос:
GOAL
родитель(X, иван), родитель(Y, иван), X<>Y.

Результат:
X=елена, Y=сергей
X=сергей, Y=елена
2 Solutions

5. Для определения всех родителей и их детей, запишите:
GOAL
родитель(X, Y).

Результат:
X=илья, Y=марина
X=марина, Y=ира
X=елена, Y=иван
X=николай, Y=ира
X=ольга, Y=алексей
X=марина, Y=саша
X=сергей, Y=иван
7 Solutions

Пример 2
Имеются факты вида: *родитель(имя, имя)* и *женщина(имя).*
а) составить правило ***мать*** и определить, кто мать Маши.
Решение:

```
DOMAINS
имя=string
PREDICATES
родитель(имя, имя)
женщина(имя)
мать(имя,имя)
CLAUSES
родитель("Марина","Ирина").
родитель("Елена", "Анна").
родитель("Ольга","Марина").
родитель("Ольга","Татьяна").
родитель("Татьяна","Катя").
родитель("Анна", "Маша").
женщина("Ольга").
женщина("Маша").
женщина("Ирина").
женщина("Елена").
женщина("Анна").
женщина("Марина").
```

```
женщина("Татьяна ").
женщина("Катя").
мать(X,Y):-родитель(X,Y),женщина(X).
GOAL
мать(X,"Маша").
```

Результат:

```
X=Анна
1 Solution
```

b) составить правило ***бабушка*** и определить, кто бабушка Маши.

Решение:

```
DOMAINS
имя=string
PREDICATES
nondeterm родитель(имя,имя)
женщина(имя)
nondeterm мать(имя,имя)
nondeterm бабушка(имя,имя)
CLAUSES
родитель("Марина","Ирина").
родитель ("Елена", "Анна").
родитель("Ольга","Марина").
родитель("Ольга","Татьяна").
родитель("Татьяна","Катя").
родитель ("Анна", "Маша").
женщина("Ольга").
женщина( "Маша").
женщина("Ирина").
женщина("Елена").
женщина("Анна").
женщина("Марина").
женщина("Татьяна ").
женщина("Катя").
мать(X,Y):-родитель(X,Y),женщина(X).
бабушка(X,Z):-мать(X,Y),родитель(Y,Z).
GOAL
бабушка(X,"Маша").
```

Результат:

```
X=Елена
1 Solution
```

Замечание: ключевое слово *nondeterm* определяет недетерминированные предикаты, которые могут совершать откат назад и генерировать множественные решения. Таким образом, если задача предполагает возможность получения несколько решений, следует объявлять предикаты как недетерминированные.

c) составить правило ***внучка*** и определить, сколько внучек у Ольги и как их зовут.

Решение:

```
DOMAINS
имя=string
PREDICATES
nondeterm родитель(имя,имя)
женщина(имя)
nondeterm мать(имя,имя)
nondeterm бабушка(имя,имя)
nondeterm внучка(имя,имя)
CLAUSES
    родитель("Марина","Ирина").
    родитель ("Елена", "Анна").
```

```
родитель("Ольга","Марина").
родитель("Ольга","Татьяна").
родитель("Татьяна","Катя").
   родитель("Анна", "Маша").
   женщина("Ольга").
женщина( "Маша").
   женщина("Ирина").
женщина("Елена").
женщина("Анна").
женщина("Марина").
женщина("Татьяна ").
женщина("Катя").
мать(X,Y):-родитель(X,Y),женщина(X).
   бабушка(X,Z):-мать(X,Y),родитель(Y,Z).
   внучка(X,Y):-бабушка(Y,X),женщина(X).
GOAL
внучка(X, "Ольга").
```

Результат:

X=Ирина
X=Катя
2 Solutions

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Имеется база данных, содержащая следующие факты:

играет (“Саша”, футбол).
играет (“Катя”, теннис).
играет (“Саша”, теннис).
играет (“Андрей”, футбол).
играет (“Олег”, футбол).
играет (“Ольга”, теннис).
играет (“Катя”, волейбол).
играет (“Олег”, волейбол).
женщина(“Катя”).
женщина(“Ольга”).
мужчина(“Саша”).
мужчина(“Андрей”).
мужчина(“Олег”).

a) используя имеющиеся факты, составить новое правило ***волейбол_жен(X)*** и определить всех женщин, играющих в волейбол;

b) используя имеющиеся факты, составить новое правило ***футбол_муж(X)*** и определить всех мужчин, играющих в футбол;

c) используя имеющиеся факты, составить правило ***теннис_пара(X,Y),*** позволяющее найти смешанную теннисную пару (мужчина+женщина). Определить все такие пары.

Создание простейших проектов

Создание проекта позволяет протестировать пример как автономную исполняемую программу. После запуска проекта на исполнение создается exe-файл, работа которого завершается после *первого* решения, удовлетворяющего решению задачи. Запуск программы в этом режиме не обеспечивает автоматический вывод значений переменных, поэтому необходимо использовать стандартный предикат вывода **write**.

Пример.

Заданы отношения-факты:

родитель(“Иван”,”Катя”).
родитель(“Анна”,” Олег ”).
родитель(“Олег”,”Дима”).
родитель(“Игорь”,”Ольга”).
родитель(“Олег”,”Виктор”).

родитель(“Игорь”,”Иван”).
мужчина(“Дима”).
мужчина(“Иван”).
мужчина(“Игорь”).
мужчина(“Олег”).
мужчина(“Виктор”).
женщина(“Катя”).
женщина(“Ольга”).
женщина(“Анна”).

Составить новое отношение-правило ***дед(X,Y)*** и определить, кто является дедушкой Кати. Создать проект и протестировать пример как автономную исполняемую программу.

Решение

1. Запустите среду Visual Prolog и создайте новый проект (Project | New Project), активизируется окно **Application Expert** (эксперт приложения).

2. Определите имя проекта (Primer) и базовый каталог, куда будет сохранен проект (например, D:\VP\ Primer)

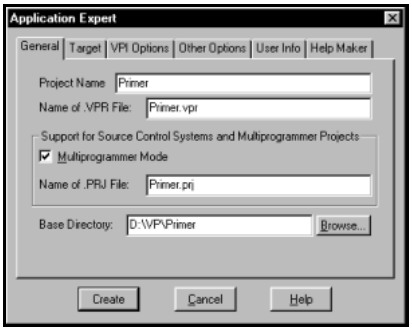


Рис.4. Окно **Application Expert**

На вкладке **Target** установите в поле UI Strategy параметр Easywin и нажмите кнопку **Create** для создания проекта (рис.5):

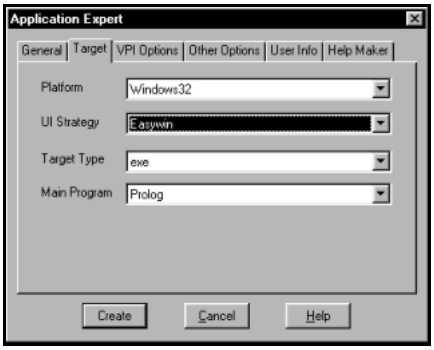


Рис.5. Установки на вкладке **Target** окна **Application Expert**

3. Откройте окно **Compiler Options** (Options | Project | Compiler Options), откройте вкладку Warnings и установите опции компилятора для созданного проекта как указано на рисунке (рис.6):

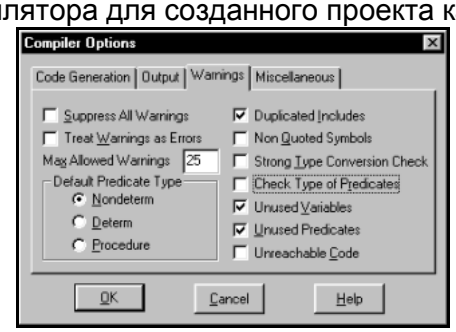


Рис.6. Установки опций компилятора

Нажмите OK.

4. В окне проекта выделите файл Primer.pro и откройте его для редактирования (двойной щелчок или кнопка **Edit**)

Файл с расширением .pro содержит секции PREDICATES, GOAL, CLAUSES. Допишите необходимые определения так, чтобы получилась программа:

DOMAINS

```
имя=string
PREDICATES
родитель(имя,имя)
женщина(имя)
мужчина(имя)
дед(имя, имя)
CLAUSES
родитель("Иван","Катя").
родитель("Анна","Олег").
родитель("Олег","Дима").
родитель("Игорь","Ольга").
родитель("Олег","Виктор").
родитель("Игорь","Иван").
мужчина("Дима").
мужчина("Иван").
мужчина("Игорь").
мужчина("Олег").
мужчина("Виктор").
женщина("Катя").
женщина("Ольга").
женщина("Анна").
дед(X,Z):-родитель(X,Y), родитель(Y,Z),
мужчина(X).
GOAL
дед(X,"Катя"),write(X).
```

5. Откомпилируйте исходный код примера и запустите его как автономную исполняемую программу. (Project | Run, или клавиша <F9>, или кнопка <R>). Результат выполнения программы должен отобразиться в окне:

Рис.7. Окно вывода результата

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

Доработайте исходный код примера следующим образом:

1) добавьте новое правило ***бабушка*** и определите, кто является бабушкой;
2) добавьте новое правило ***внук*** и определите, кто внук Анны;
3) добавьте новое правило ***брат*** и определите, кто брат Димы;
4) добавьте новое правило ***сестра*** и определите, кто сестра Ивана.

Лабораторная работа №2
Поиск с возвратом

Поиск с возвратом (backtracking) – это один из основных приемов поиска решений поставленной задачи в ПРОЛОГе. Выполняя поиск, ПРОЛОГ может столкнуться с необходимостью выбора между альтернативными путями. Тогда он ставит маркер у места развилки (точка отката) и выбирает первую подцель. Если она не выполняется, то ПРОЛОГ возвращается в точку отката и переходит к следующей подцели.

Среда Visual Prolog позволяет использовать отладчик для пошагового выполнения программы. Отладчик работает с откомпилированным кодом. В исходном коде можно ставить точки останова и выполнять программу по шагам. В режиме пошагового выполнения программы можно просматривать значения переменных и содержимое утвержденных фактов.

Пример

Имеется база данных, содержащая факты вида ***отдыхает(имя, город)***, ***украина(город***), ***россия(город), прибалтика(город).*** Составить правило, позволяющее определить, кто отдыхал в России.

Проследить поиск решения задачи с помощью отладчика Visual Prolog и построить целевое дерево поиска с возвратом.

Решение:

1. Создайте новый проект (Project | New Project) и наберите текст программы:

```
DOMAINS
имя, город=string
PREDICATES
отдыхает(имя, город)
украина(город)
россия(город)
прибалтика(город)
отдых_Россия(имя)
CLAUSES
отдыхает(sasha, antalia).
отдыхает(anna, sochi).
отдыхает(dima, urmala).
отдыхает(oleg, kiev).
украина(kiev).
россия(sochi).
прибалтика(urmala).
отдых_Россия(X):- отдыхает(X,Y),россия(Y).
GOAL
отдых_Россия(X),
write(X),nl.
```

3. Сохраните проект (**Project** | **Save Project**)

4. Запустите его на исполнение (**Project** | **Run**, или клавиша **<F9>,** или кнопка **<R>**).

Результат выполнения программы:

anna

5. Проследите поиск этого решения с помощью отладчика(Debugger). Для этого:

а) запустите отладчик (**Project** | **Debug**);

б) в окне отладчика выберите команду **View** | **Local Variables** (для просмотра текущих значений переменных);

в) нажимайте клавишу **<F7>** (или **Run** | **Trace Into**) для пошагового выполнения программы, текущие значения переменных отображаются в окне Variables For Current Clause

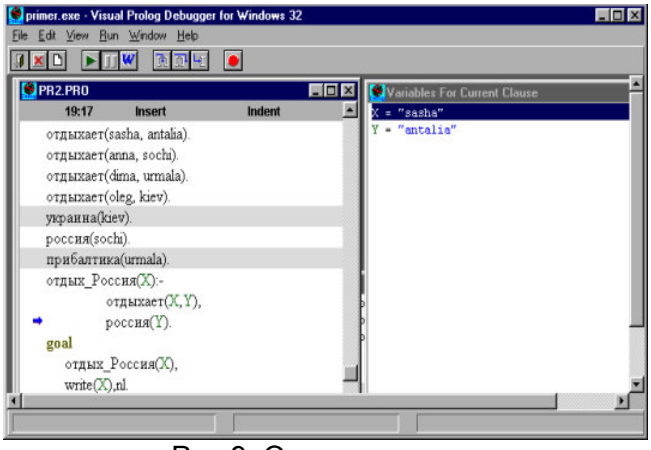


Рис.9. Окно отладчика

Поиск решения можно представить следующим образом:

Рис.10. Целевое дерево поиска решения

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. База данных содержит следующие факты:

увлекается(“Коля”, гитара).
увлекается(“Оля”, скрипка).
увлекается(“Дима”, плаванье).
увлекается(“Таня”, теннис).
спорт(плаванье).
спорт(теннис).
муз_инстр(скрипка).
муз_инстр(гитара).

а) составить правило спортсмен и определить, кто увлекается спортом;
б) проследить за поиском решения с помощью отладчика;
в) построить целевое дерево поиска с возвратом.

Управление поиском с возвратом: предикаты fail и отсечения.

fail – это тождественно-ложный предикат, искусственно создающий ситуацию неуспеха. После выполнения этого предиката управление передается в точку отката и поиск продолжается. Использование предиката fail позволяет найти все решения задачи.

Чтобы ограничить пространство поиска и прервать поиск решений при выполнении какого-либо условия, используется предикат *отсечения* (обозначается **!**), Однажды пройдя через отсечение, невозможно вернуться назад, т.к. этот предикат является тождественно-истинным. Процесс может только перейти к следующей подцели, если такая имеется.

Например, p :- p1, p2, !, p3.

Если достигнуты цели p1 и p2, то возврат к ним для поиска новых решений невозможен.

Пример 1

База данных содержит факты вида: ***student(имя, курс).*** Создать проект, позволяющий сформировать список студентов 1-го курса.

Решение:

```
PREDICATES
student(symbol,integer)
spisok
CLAUSES
student(vova,3).
student(lena,1).
student(dima,1).
student(ira,2).
student(marina,1).
spisok:-student(X,1),write(X),nl,fail.
GOAL
write("Список студентов 1-курса"),nl,spisok.
```

Результат выполнения программы:

Список студентов 1-курса

lena
dima
marina

Пример 2

База данных содержит факты вида **father(name, name).** Создать проект, позволяющий определить кто чей отец.

Решение:

```
DOMAINS
name=symbol
PREDICATES
father (name, name)
everybody
CLAUSES
father ("Павел", "Петр").
father ("Петр", "Михаил").
father ("Петр", "Иван").
everybody:- father (X, Y),
write(X," - это отец",Y,"a"),nl,
fail.
GOAL
everybody.
```

Результат выполнения программы:

Павел - это отец Петра
Петр - это отец Михаила
Петр - это отец Ивана

Пример 3

Создать проект, реализующий железнодорожный справочник. В справочнике содержится следующая информация о каждом поезде: номер поезда, пункт назначения и время отправления.

а) вывести всю информацию из справочника.

Решение:

```
DOMAINS
nom=integer
p, t=string
PREDICATES
poezd(nom,p,t)
CLAUSES
poezd(233,moskva,"12-30").
poezd(257,moskva,"22-40").
poezd(133,armavir,"10-20").
poezd(353,armavir,"20-40").
poezd(353,adler,"02-30").
poezd(413,adler,"11-10").
poezd(256,piter,"21-30").
GOAL
write(" Расписание поездов"), nl,
write("Номер Пункт прибытия Время отправления"),
nl, poezd(N,P,T), write(N," ",P," ",T),nl,fail.
```

Результат выполнения программы

Расписание поездов
Номер Пункт прибытия Время отправления
233 moskva 12-30
257 moskva 22-40
133 armavir 10-20
353 armavir 20-40
353 adler 02-30

413 adler 11-10
256 piter 21-30

б) организовать поиск поезда по пункту назначения.

Решение:

```
GOAL
write (" Пункт назначения:"), Readln(P), nl,
write ("Номер Время отправления"), nl,
poezd(N,P,T), write(N," ",T), nl, fail.
```

Комментарий: Readln –стандартный предикат ввода строкового значения

Результат выполнения программы

Пункт назначения:armavir

Номер Время отправления
133 10-20
353 20-40

в) вывести информацию о поездах, отправляющихся в заданный временной промежуток

Решение:

```
GOAL
write(" Время отправления:"),nl,
write("с..."), Readln(T1),
write("до..."), Readln(T2), nl,
write("Номер Пункт назначения Время отправления"),
nl,poezd(N,P,T),T>=T1,T<=T2,write(N," ",P," ", T),
nl, fail.
```

Результат выполнения программы

Время отправления:
с...10-00
до...15-00

Номер Пункт назначения Время отправления
233 moskva 12-30
133 armavir 10-20
413 adler 11-10

Пример 4

Имеется база данных, содержащая данные о спортсменах: имя и вид спорта. Определить возможные пары одного из спортсменов-теннисистов с другими теннисистами.

Решение:

```
DOMAINS
имя,вид_сп=string
PREDICATES
играет(имя,вид_сп)
спис_спортс
CLAUSES
играет("Саша",теннис).
играет("Аня",волейбол).
играет("Олег",футбол).
играет("Коля",теннис).
играет("Саша",футбол).
играет("Андрей",теннис).
спис_спортс:- играет(X,теннис),!,играет(Y,теннис),
X<>Y,write(X,"-",Y),nl,fail.
GOAL
write("Пары теннисистов"),nl,
спис_спортс.
```

Результат выполнения программы:

Пары теннисистов
Саша-Коля
Саша-Андрей

Пример 5

Студенту в зависимости от набранной в процессе обучения суммы баллов **Z** присваивается квалификация:

магистр (**M**), если 80<=Z<=100
специалист (**S**), если 60<= Z< 80
бакалавр (**B**), если 40<= Z< 60
неудачник (**N**), если 0<=Z< 40

Составить программу, которая определит квалификацию в зависимости от введенного значения **Z**

Решение:

Для решения задачи составим правило ***grade***, устанавливающее связь между количеством баллов (z) и квалификацией (r). Правило состоит из нескольких частей. Первые две части обеспечивают проверку недопустимых значений Z с выводом соответствующего сообщения. Остальные части правила определяют квалификацию в зависимости от значения Z.

```
DOMAINS
z=integer
r=string
PREDICATES
grade(z,r)
CLAUSES
grade(Z,""):-Z<0,!, write("Неверный ввод данных!").
grade(Z,""):-Z>100,!,write("Неверный ввод данных!").
grade(Z,"M"):-Z>=80,!.
grade(Z,"S"):-Z>=60,!.
grade(Z,"B"):-Z>=40,!.
grade(Z,"N").
GOAL
write("Z="), readint(Z), grade(Z,R),write(R).
```

Комментарий: readint – стандартный предикат ввода целочисленного значения

Результат выполнения программы:

1-й случай:
Z=88
M
2-й случай:
Z=65
S
3-й случай:
Z=39
N
4-й случай:
Z=110
Неверный ввод данных!

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. База данных содержит факты вида: **отдыхает(имя, город), украина(город), россия(город), женщина (имя), мужчина(имя).**

а) вывести список женщин, отдыхающих в России;

б) вывести список мужчин, отдыхающих на Украине.

2. База данных содержит факты вида: **книга(автор, название, издательство, год_издания), украина(город).**

а) вывести весь список книг;

б) вывести список книг авторов Пушкина и Чехова;

в) вывести список книг, изданных в издательстве «Питер» не ранее 2000 года.

3. Составить программу, реализующую авиасправочник. В справочнике содержится следующая информация о каждом рейсе: номер рейса, пункт назначения, время вылета, дни(ежедн., чет, нечет). Вывести:

а) всю информацию из справочника;

б) информацию о самолетах, вылетающих в заданный пункт по четным дням;

в) информацию о самолетах, вылетающих ежедневно не позже указанного времени.

4. Составить программу, реализующую географический справочник. В справочнике содержится следующая информация о каждой стране: название страны, название столицы, численность населения, географическое положение (Европа или Азия). Вывести:

а) всю информацию из справочника;

б) информацию о странах, численность населения которых превышает заданное значение;

в) информацию о европейских странах, численность населения которых не превышает заданное значение.

5. Составить программу, реализующую словарь. В словаре содержится следующая информация: слово и его перевод (русские и английские слова). Реализовать вывод всего словаря, перевод с русского на английский, с английского на русский (с несколькими значениями).

6. Составить программу, реализующую телефонный справочник. В справочнике содержится следующая информация о каждом абоненте: имя и телефон. Реализовать вывод всей информации из справочника, поиск телефона по имени, поиск имени по телефону

7. База данных содержит факты вида: ***ученик(имя, класс)*** и ***увлекается(имя, хобби).*** Составить программу, которая выводит:

а) список всех учеников и их увлечения;

б) подбирает одному из учеников указанного класса, увлекающемуся кино, пару из других классов. Вывести все возможные пары.

8. База данных содержит факты вида: ***ученик(имя, класс)*** и ***играет(имя, вид_спорта).*** Составить программу, которая:

а) выводит список всех учеников заданного класса и вид спорта, которым они увлекаются;

б) подбирает одному из учеников указанного класса, играющему в бадминтон, пару из других классов. Вывести все возможные пары.

Лабораторная работа №3
Арифметические вычисления

Хотя Пролог не предназначен для решения вычислительных задач, его возможности вычислений аналогичны соответствующим возможностям таких языков программирования как Basic, C, Pascal.

В языке Пролог имеется ряд встроенных функций для вычисления арифметических выражений, некоторые из которых перечислены в таблице 1.

Таблица 1. Математические операции и функции в Прологе

X + Y	Сумма X и Y
X – Y	Разность X и Y
X * Y	Произведение X и Y
X / Y	Деление X на Y
X mod Y	Остаток от деления X на Y
X div Y	Целочисленное деление X на Y
abs(X)	Абсолютная величина числа X
sqrt(X)	Квадратный корень из X
random(X)	Случайное число в диапазоне от 0 до 1
random(Int,X)	Случайное целое число в диапазоне от 0 до Int
round(X)	Округление X
trunc(X)	Целая часть X
sin(X)	Синус X
cos(X)	Косинус X

arctan(X)	Арктангенс X
tan(X)	Тангенс X
ln(X)	Натуральный логарифм X
log(X)	Логарифм X по основанию 10

Пример 1.

Вычислить значение выражения Z=(2*X+Y)/(X-Y) для введенных X и Y.

Решение:

```
PREDICATES
        знач_выраж(real,real)
CLAUSES
        знач_выраж(X,Y):-X<>Y, Z=(2*X+Y)/(X-Y),
        write("Z=",Z);
        X=Y, write ("Делить на 0 нельзя!").
GOAL
        Write("X="),readreal(X),
        Write("Y="),readreal(Y),знач_выраж(X,Y),nl.
```

Комментарий: readreal – предикат для ввода действительных чисел

Результат выполнения программы:

1-й случай:

```
        X=4
        Y=4
        Делить на 0 нельзя!
```

2-й случай:

```
        X=5
        Y=2
        Z=4
```

Пример 2.

Найти минимальное из двух введенных A и B.

Решение:

```
PREDICATES
        min(integer,integer,integer)
CLAUSES
        min(A,B,A):-A<=B,!.
        min(A,B,B).
GOAL
        Write("A="),readreal(A),Write("B="),readreal(B),
        min(A,B,Min),write("min=",Min),nl.
```

Результат выполнения программы:

1-й случай:

```
        A=5
        B=17
        min=5
```

2-й случай:

```
        A=35
        B=18
        min=18
```

3-й случай:

```
        A=8
        B=8
        min=8
```

Пример 3.

Определить, является четным или нечетным случайным образом выбранное число от 0 до 20.

Решение:

```
PREDICATES
        chet
CLAUSES
        chet:-random(20,X),write(X),X mod 2=0,
        write(" - четное"),!.
        chet:-write( " - нечетное").
GOAL
        chet.
```

Результат выполнения программы:

1-й случай:

6 – четное

2-й случай:

19 – нечетное

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Составить программу для вычисления значения выражения Y=(X2+1)/(X-2) для введенного X.
2. Составить программу для вычисления значения выражения S=2(X2+Y2)/(X+Y) для введенных X и Y.
3. Составить программу для вычисления значения выражения $z=e^x \sin x + 3\ln x$ для введенного X.
4. Составить программу для вычисления значения выражения $y=\ln(\lg(\sin x+e^x))$для введенного X.
5. Составить программу для вычисления среднего арифметического двух введенных чисел.
6. Составить программу для вычисления среднего геометрического двух введенных чисел.
7. Составить программу для проверки введенного натурального числа на четность.
8. Составить программу для проверки попадает ли введенное число X в заданный промежуток [a,b].
9. Составить программу для выбора наименьшего из трех введенных чисел.
10. Составить программу для выбора наибольшего из трех введенных чисел.

Лабораторная работа №4
Рекурсия

Рекурсивная процедура – это процедура, вызывающая сама себя до тех пор, пока не будет соблюдено некоторое условие, которое остановит рекурсию. Такое условие называют *граничным*. Рекурсивное правило всегда состоит по крайней мере из двух частей, одна из которых является *нерекурсивной*. Она и определяет граничное условие.

В рекурсивной процедуре нет проблемы запоминания результатов ее выполнения, потому что любые вычисленные значения можно передавать из одного вызова в другой как аргументы рекурсивно вызываемого предиката. Рекурсия является эффективным способом для решения задач, содержащих в себе подзадачу такого же типа.

Пример 1.

База данных содержит следующие факты:

```
roditel(ivan,oleg).
roditel(inna,oleg).
roditel(oleg,dima).
roditel(oleg,marina).
```

Составить рекурсивное правило ***предок*** и определить всех предков и их потомков.

Решение:

```
DOMAINS
        name=string
PREDICATES
        roditel(name,name)
        predok(name,name)
CLAUSES
```

```
        roditel(ivan,oleg).
        roditel(inna,oleg).
        roditel(oleg,dima).
        roditel(oleg,marina).
        predok(X,Z):-roditel(X,Z).    % нерекурсивная часть правила
        predok(X,Z):-roditel(X,Y),    % рекурсивная часть правила
                predok(Y,Z).
GOAL
        predok(X,Y),
        write("Predok -",X," Ego potomok-",Y),nl,fail.
```

Результат выполнения программы:

```
        Predok -ivan   Ego potomok-oleg
        Predok -inna   Ego potomok-oleg
        Predok -oleg   Ego potomok-dima
        Predok -oleg   Ego potomok-marina
        Predok -ivan   Ego potomok-dima
        Predok -ivan   Ego potomok-marina
        Predok -inna   Ego potomok-dima
        Predok -inna   Ego potomok-marina
```

Пример 2. Вычисление факториала.
Решение:

```
PREDICATES
        fact(integer,integer)
CLAUSES
        fact(0,1):-!.              % Факториал нуля равен единице
        fact(N,F):- N1=N-1,        %   уменьшаем N на единицу,
                fact(N1,F1),       % вычисляем факториал нового числа,
                F=N*F1.            %  а затем умножает его на N
GOAL
        write("N="),readint(N),fact(N,F),write("F=",F),nl.
```

Результат выполнения программы:

```
1-й случай:
        N=0
        F=1
2-й случай:
        N=1
        F=1
3-й случай:
        N=4
        F=24
```

Пример 3. Составить программу для вычисления $Y=X^n$, X, n – целые числа
Решение:

Составим правило **stepen**, состоящее из 3-х частей.

1-я часть правила (нерекурсивная) определяет, что $X^0=1$.

2-я часть правила (рекурсивная) вычисляет X^n для положительного n.

3-я часть (рекурсивная) - вычисляет X^n для отрицательного n (добавляется необходимое условие X<>0)

```
PREDICATES
        stepen(real,real,real)
CLAUSES
        stepen(X,0,1):-!.
        stepen(X,N,Y):-N>0,N1=N-1,stepen(X,N1,Y1),Y=Y1*X,!.
        stepen(X,N,Y):-X<>0,K=-N,stepen(X,K,Z),Y=1/Z.
```

```
GOAL
        write("X="),readreal(X),
        write("N="),readreal(N),
        stepen(X,N,Y),write("Y=",Y),nl.
```

Результат выполнения программы:

1-й случай:

```
        X=3
        N=2
        Y=9
```

2-й случай:

```
        X=2
        N=-2
        Y=0.25
```

Пример 4. Ханойские башни

Имеется три стержня: A, B и C. На стержне A надеты N дисков разного диаметра, надетые друг на друга в порядке убывания диаметров. Необходимо переместить диски со стержня A на стержень C используя B как вспомогательный, если перекладывать можно только по одному диску и нельзя больший диск класть на меньший.

Решение:

Составим правило **move**, определяющее порядок переноса дисков.

1-я (нерекурсивная) часть правила определяет действие, если на стержне находится 1 диск.

2-я (рекурсивная) часть правила перемещает сначала верхние N-1 диск на стержень B, используя C как вспомогательный, затем оставшийся диск на стержень C и, наконец, диски со стержня B на C, используя A как вспомогательный.

```
PREDICATES
        move(integer,char,char,char)
CLAUSES
        move(1,A,B,C):-
                write("Перенести диск с ",A," на  ",C),nl,!.
        move(N,A,B,C):-
                M=N-1,move(M,A,C,B),
                write("Перенести диск с ",A," на  ",C),nl,
                move(M,B,A,C).
GOAL
        write("Ханойские башни"), nl,
        write("Количество дисков:"), readint(N),nl,
        move(N,'A','B','C').
```

Результат выполнения программы:

```
        Ханойские башни
        Количество дисков:3

        Перенести диск с A на  C
        Перенести диск с A на  B
        Перенести диск с C на  B
        Перенести диск с A на  C
        Перенести диск с B на  A
        Перенести диск с B на  C
        Перенести диск с A на  C
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Вычислить сумму 1+2+3+…+N.
2. Подсчитать сумму ряда целых четных чисел от 2 до N.
3. Вычислить сумму ряда целых нечетных чисел от 1 до n.
4. Найти значение произведения: 2*4*6*...*26
5. Найти значение произведения: 1*3*5*...*11

6. Вычислить значение n-го члена ряда Фибоначчи: f(0)=0, f(1)=1, f(n)=f(n-1)+f(n-2).
7. Используя базу данных и правило **предок** из примера 2 составить правило для определения всех потомков-мужчин.
8. Используя базу данных и правило **предок** из примера 2 составить правило для определения всех потомков-женщин.

Лабораторная работа №5
Решение логических задач в Prolog

ПРОЛОГ позволяет наиболее естественным образом решать логические задачи, моделируя процесс размышления человека с помощью правил.

Многие логические задачи связаны с рассмотрением нескольких конечных множеств с одинаковым количеством элементов, между которыми устанавливается взаимно-однозначное соответствие. В ПРОЛОГе эти множества можно описывать как базы данных, а зависимости между объектами устанавливать с помощью правил.

Пример 1

В автомобильных гонках три первых места заняли Алеша, Петя и Коля. Какое место занял каждый из них, если Петя занял не второе и не третье место, а Коля - не третье?

Решение

Традиционным способом задача решается заполнением таблицы. По условию задачи Петя занял не второе и не третье место, а Коля - не третье. Это позволяет поставить символ '-' в соответствующих клетках.

Имя	I место	II место	III место
Алеша			
Петя		-	-
Коля			-

Между множеством имен участников гонки и множеством мест должно быть установлено взаимнооднозначное соответствие. Поэтому определяем занятое место сначала у Пети, затем у Коли и, наконец, у Алеши. В соответствующих клетках проставляем знак '+'. В каждой строке и каждом столбце должен быть только один такой знак.

Имя	I место	II место	III место
Алеша	**-**	**-**	**+**
Петя	**+**	**-**	**-**
Коля	**-**	**+**	**-**

Из последней таблицы следует, что Алеша занял третье место, Петя - первое, Коля - второе.

Программа на ПРОЛОГе будет выглядеть следующим образом:

```
PREDICATES
        имя(string)
        место(string)
        соответствие(string,string)
        решение(string,string,string,string,string,string)
CLAUSES
        имя(алеша).
        имя(петя).
        имя(коля).
        место(первое).
        место(второе).
        место(третье).
        /* Устанавливаем взаимнооднозначное соответствие
        между множеством имен и множеством мест, X - имя,Y -  место */
        /* Петя занял не второе и не третье место */
        соответствие(X, Y):-имя(X), X=петя,
                        место(Y),not(Y=второе),
```

```
                    not(Y=третье).
        /* Коля занял не третье место */
        соответствие(X, Y):- имя(X), X=коля,
                        место(Y), not(Y=третье).
        соответствие(X, Y):- имя(X),  X=алеша, место(Y).
        /* У всех ребят разные места */
        решение(X1,Y1,X2,Y2,X3,Y3):-
                        X1=петя,соответствие(X1,Y1),
                        X2=коля,соответствие(X2,Y2),
                        X3=алеша,соответствие(X3,Y3),
                        Y1<>Y2, Y2<>Y3, Y1<>Y3.
GOAL
        решение(X1,Y1,X2,Y2,X3,Y3), write(X1,"  -  ",Y1),nl,
        write(X2,"  -  ",Y2),nl,write(X3,"  -  ",Y3),nl.
```

Результат выполнения программы

```
петя  -  первое
коля  -  второе
алеша  - третье
```

Пример 2. Наташа, Валя и Аня вышли на прогулку, причем туфли и платье каждой были или белого, или синего, или зеленого цвета. У Наташи были зеленые туфли, а Валя не любит белый цвет. Только у Ани платье и туфли были одного цвета. Определить цвет туфель и платья каждой из девочек, если у всех туфли и платья были разного цвета.

Решение

```
PREDICATES
        имя(string)
        туфли(string)
        платье(string)
        соот(string,string,string)
        решение(string,string,string,string,string,string,
              string,string,string)
CLAUSES
        имя(наташа).
        имя(валя).
        имя(аня).
        туфли(белый).
        туфли(синий).
        туфли(зеленый).
        платье(белый).
        платье(синий).
        платье(зеленый).
        % X – имя,  Y – цвет туфель, Z – цвет платья
        соот(X,Y,Z):-имя(X),туфли(Y),платье(Z),
                  X=наташа,Y=зеленый,Y<>Z.
        соот(X,Y,Z):-имя(X),туфли(Y),платье(Z),
                  X=валя,not(Y=белый),
                  not(Z=белый), Y<>Z.
        соот(X,Y,Z):-имя(X),туфли(Y),платье(Z),X=аня,Y=Z.
        решение(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3):-
                        X1=наташа,соот(X1,Y1,Z1),
                        X2=валя, соот(X2,Y2,Z2),
                        X3=аня, соот(X3,Y3,Z3),
                        Y1<>Y2, Y2<>Y3, Y1<>Y3,
                        Z1<>Z2, Z2<>Z3, Z1<>Z3.
GOAL
        решение(X1,Y1,Z1,X2,Y2,Z2,X3,Y3,Z3),
        write(X1,"  туфли-  ",Y1," платье- ",Z1),nl,
```

```
write(X2,"  туфли-  ",Y2," платье- ",Z2),nl,
write(X3,"  туфли-  ",Y3," платье- ",Z3),nl.
```

Результат выполнения программы

```
наташа   туфли -  зеленый  платье - синий
валя     туфли -  синий    платье - зеленый
аня      туфли -  белый    платье - белый
```

Пример 3. Витя, Юра и Миша сидели на скамейке. В каком порядке они сидели, если известно, что Миша сидел слева от Юры, а Витя слева от Миши.

Решение

```
PREDICATES
        слева(string,string)
        ряд(string,string,string)
CLAUSES
        /* Миша сидел слева от Юры */
        слева(миша, юра).
        /* Витя сидел слева от Миши */
        слева(витя, миша).
        /* Объекты X, Y и Z образуют ряд,
        если X слева от Y и Y слева от Z */
        ряд(X, Y, Z):- слева(X,Y), слева(Y, Z).
GOAL
        ряд(X, Y, Z), write(X,"-",Y,"-",Z),nl.
```

Результат выполнения программы

```
витя-миша-юра
```

Пример 4. Известно, что тополь выше березы, которая выше липы. Клен ниже липы, а сосна выше тополя и ниже ели. Определить самое высокое и самое низкое дерево.

Решение

```
DOMAINS
        name=string
PREDICATES
        выше(name,name)
        ряд(name,name,name,name,name,name)
CLAUSES
        выше(тополь,береза).
        выше(липа,клен).
        выше(ель,сосна).
        выше(береза,липа).
        выше(сосна,тополь).
        ряд(X1,X2,X3,X4,X5,X6):-выше(X1,X2),выше(X2,X3),
                        выше(X3,X4),выше(X4,X5),
                        выше(X5,X6).
GOAL
        ряд(X,_,_,_,_,Y),write("Самое высокое - ",X),nl,
        write("Самое низкое - ",Y),nl.
```

Результат выполнения программы

```
Самое высокое - ель
Самое низкое - клен
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Трое ребят вышли гулять с собакой, кошкой и хомячком. Известно, что Петя не любит кошек и живет в одном подъезде с хозяйкой хомячка. Лена дружит с Таней, гуляющей с кошкой. Определить, с каким животным гулял каждый из детей.

2. Беседуют трое друзей: Белокуров, Рыжов и Чернов. Брюнет сказал Белокурову: «Любопытно, что один из нас блондин, другой – брюнет, а третий – рыжий, но ни у кого цвет волос не соответствует фамилии». Какой цвет волос у каждого из друзей?
3. Витя, Юра, Миша и Дима сидели на скамейке. В каком порядке они сидели, если известно, что Юра сидел справа от Димы, Миша справа от Вити, а Витя справа от Юры.
4. Известно, что Волга длиннее Амударьи, а Днепр короче Амударьи. Лена длиннее Волги. Определить вторую по протяженности реку.

Лабораторная работа №6
Списки

Список – это объект, который содержит конечное число других объектов. Список в ПРОЛОГе заключается в квадратные скобки и элементы списка разделяются запятыми. Список, который не содержит ни одного элемента, называется *пустым* списком.

Список является рекурсивным объектом. Он состоит из *головы* (первого элемента списка) и *хвоста* (все последующие элемента). Хвост также является списком. В ПРОЛОГе имеется операция "|", которая позволяет делить список на голову и хвост. Пустой список нельзя разделить на голову и хвост.

Тип данных "список" объявляется в программе на Прологе следующим образом:

```
DOMAINS
        списковый_тип = тип*
```

где "тип" - тип элементов списка; это может быть как стандартный тип, так и нестандартный, заданный пользователем и объявленный в разделе DOMAINS ранее.

- формирование списка;
- объединение списков;
- поиск элемента в списке;
- вставка элемента в список и удаление из списка.

Основными операциями на списками являются:

Пример 1. Сформировать список вида [7,6,5,4,3,2,1]

Решение

```
DOMAINS
        list = integer*
PREDICATES
        genl(integer, list)
CLAUSES
        genl(0,[]):-!.
        genl(N,[N|L]):-N1=N-1, genl(N1,L).
GOAL
        genl(7,L),write(L),nl.
```

Результат выполнения программы:

```
        [7,6,5,4,3,2,1]
```

Пример 2. Сформировать список из N элементов, начиная с 2. Каждый следующий на 4 больше предыдущего.

Решение

```
DOMAINS
        list = integer*
PREDICATES
        genl( integer, integer, list )
CLAUSES
        genl(N2,N2,[]):-!.
        genl(N1,N2,[N1|L]):-N1<N2, N=N1+4,
                genl(N,N2,L).
GOAL
        write("N="),readint(N),K=4*(N+1)-2,
   genl(2,K,L),write(L),nl.
```

Результат выполнения программы:

N=5
[2,6,10,14,18]

Пример 3. Сформировать список последовательных натуральных чисел от 4 до 20 и найти количество его элементов.

Решение:

```
DOMAINS
        list = integer*
PREDICATES
        genl1(integer, integer, list )
        len(integer, list )
CLAUSES
        genl1(N2,N2,[]):-!.
        genl1(N1,N2,[N1|L]):-N1<N2, N=N1+1, genl1(N,N2,L).
        len(0,[]).
        len(X,[_|L]):-len(X1,L), X=X1+1.
GOAL
        genl1 (4,21,L ),write(L),nl,
        len(X, L),write("Количество элементов=",X),nl.
```

Результат выполнения программы:

[4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]
Количество элементов=17

Пример 4. Определить, содержится ли введенное число X в заданном списке L.

Решение:

```
DOMAINS
    list = integer*
PREDICATES
    member(integer, list)
CLAUSES
    member(X,[X|_]):-write("yes"),!.
    member(X,[]):-write("no"),!.
    member(X,[_|L]) :- member(X, L).
GOAL
    L=[1,2,3,4], write(L),nl,     write("X="),readint(X),
    member(X, L),nl.
```

Результат выполнения программы:

1-й случай:

[1,2,3,4]
X=3
yes

2-й случай:

[1,2,3,4]
X=5
no

Пример 5. Сформировать списки L1=[1,2,3], L2=[10,11,12,13,14,15] и объединить их в список L3.

Решение:

```
DOMAINS
        list = integer*
PREDICATES
        genl1(integer,integer,list)
        append(list,list,list)
CLAUSES
        genl1(N2,N2,[]):-!.
        genl1(N1,N2,[N1|L]):-N1<N2,N=N1+1,genl1(N, N2, L).
        append([],L,L).
        append([X|L1],L2,[X|L3]):-append(L1,L2,L3).
```

```
GOAL
        genl1(1,4,L1),write("L1=",L1),nl,
        genl1(10,16,L2),write("L2=",L2),nl,
        append(L1,L2,L3),write("L3=",L3),nl.
```

Результат выполнения программы:

```
        L1=[1,2,3]
        L2=[10,11,12,13,14,15]
        L3=[1,2,3,10,11,12,13,14,15]
```

Пример 6. Удалить из списка, элементами которого являются названия дней недели, указанный элемент.

Решение:

```
DOMAINS
        list = symbol*
PREDICATES
        del(symbol,list,list)
CLAUSES
        del(X,[X|L],L).
        del(X,[Y|L],[Y|L1]):-del(X,L,L1).
GOAL
        L=[пн, вт, ср, чт, пт, сб, вс],write("L=",L),nl,
        write("X="),readln(X),
        del(X,L,L1),write("L1=",L1),!;
        write("Элемент отсутствует в списке"),nl.
```

Результат выполнения программы:

```
1-й случай:
        L=["пн","вт","ср","чт","пт","сб","вс"]
        X=ср
        L1=["пн","вт","чт","пт","сб","вс"]
2-й случай:
        L=["пн","вт","ср","чт","пт","сб","вс"]
        X=пр
        Элемент отсутствует в списке
```

Пример 7. Вставить в список имен новый элемент, значение которого вводится с клавиатуры. Вывести все возможные варианты вставок.

Решение:

```
DOMAINS
    list = symbol*
PREDICATES
    del(symbol,list,list)
    ins(symbol,list,list)
CLAUSES
    del(X,[X|L],L).
    del(X,[Y|L],[Y|L1]):-del(X,L,L1).
    ins(X,L1,L):-del(X,L,L1).
GOAL
    L=[olga, oksana, toma, dima],write("L=",L),nl,
    write("X="),readln(X),
    ins(X,L,L1),write("L1=",L1),nl, fail.
```

Результат выполнения программы:

```
    L=["olga","oksana","toma","dima"]
    X=vera
    L1=["vera","olga","oksana","toma","dima"]
    L1=["olga","vera","oksana","toma","dima"]
    L1=["olga","oksana","vera","toma","dima"]
    L1=["olga","oksana","toma","vera","dima"]
    L1=["olga","oksana","toma","dima","vera"]
```

Пример 8. Найти сумму элементов списка целых чисел.

Решение:

```
DOMAINS
        list=integer*
PREDICATES
        sum_list(list, integer)
CLAUSES
        sum_list([],0).
        sum_list([X|L],S):-sum_list(L,S1),S=S1+X.
GOAL
        L=[1,2,3,4,5],sum_list(L,S), write("S=",S).
```

Результат выполнения программы:

```
        S=15
```

ЗАДАНИЯ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

1. Сформировать список [2, 4, 6, 8, 10] и удалить из него введенное число.
2. Сформировать списки [1, 3, 5, 7, 9] и [2, 4, б, 8, 10] и объединить их в один.
3. Сформировать список [3, 6, 9, 12, 15, 18] и вставить в него введенное число.
4. Сформировать список из N натуральных чисел, начиная с 10. Каждое следующее на 5 больше предыдущего.
5. Сформировать список [3, 6, 9, 12, 15] и найти сумму его элементов
6. Сформировать список [6, 5, 4, 3, 2] и найти сумму его элементов
7. Сформировать список [7, 5, 3, 1] и найти произведение его элементов
8. Сформировать список из N последовательных натуральных чисел, начиная с 10. Найти сумму его элементов

19.3.8 Комплект индивидуальных заданий
Тема Поиск с возвратом

Вариант 1

Написать программу, реализующую телефонный справочник. В справочнике содержится следующая информация о каждом абоненте: имя и телефон. Реализовать вывод всей информации из справочника, поиск телефона по имени, поиск имени по телефону. Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 2

Написать программу, реализующую географический справочник. В справочнике содержится следующая информация: названия стран и площади страны. Реализовать вывод всей информации из справочника, поиск по названию. Реализовать поиск по площади, при этом должна быть возможность ввести некоторое пороговое значение (например, вывести названия всех стран, площадь которых не менее 3 млн. км^2). Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 3

Написать программу, реализующую калькулятор на четыре арифметических действия (без скобок).

Вариант 4

Написать программу, реализующую словарь. В словаре содержится следующая информация: слово и его несколько переводов. Реализовать вывод всего словаря, перевод с русского на английский, с английского на русский. Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 5

Написать программу, реализующую авиасправочник. В справочнике содержится следующая информация о каждом рейсе: номер рейса, пункт назначения, цена билета. Реализовать вывод всей информации из справочника, поиск пункта назначения по номеру рейса. Реализовать поиск по пункту назначения с указанием максимально возможной цены билета (должны быть выведены все рейсы, цена билета на которые ниже указанного значения). Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 6

Написать программу для продажи театральных билетов. Должна быть представлена следующая информация: спектакль, свободные места, цена билета. Реализовать вывод всей информации о билетах, поиск билета по ряду. Реализовать поиск по цене с указанием максимально возможной цены (должна быть выведена информация о билетах, цены на которые ниже указанного значения). Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 7

Написать программу, реализующую автомагазин. Должна быть представлена следующая информация о каждом автомобиле: модель, мощность двигателя, цвет, цена. Реализовать вывод всей информации по автомобилям, поиск по цвету. Реализовать поиск по мощности с указанием минимальной мощности (должна быть выведена информация обо всех моделях, мощность которых выше указанного значения). Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 8

Написать программу, реализующую книжный магазин. Должна быть представлена следующая информация: название книги, количество экземпляров, цена. Реализовать вывод всей информации о книгах, поиск книги по названию. Реализовать поиск по цене с указанием интервала возможной цены (должна быть выведена информация о книгах, цены которых попадают в указанный интервал). Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 9

Написать программу для продажи туристических туров. Должна быть представлена следующая информация: название тура, страна, продолжительность, цена. Реализовать вывод информации обо всех турах, поиск тура по стране. Реализовать поиск по продолжительности с указанием интервала возможной продолжительности (должна быть выведена информация о турах, продолжительность которых попадает в указанный интервал). Для удобства работы реализовать меню с соответствующими пунктами.

Вариант 10

Написать программу для заказа мест в отеле. Должна быть представлена следующая информация: название отеля, класс отеля, свободные места, цена номера. Реализовать вывод информации обо всех свободных номерах, поиск отеля по классу. Реализовать поиск по цене с указанием максимально возможной цены (должна быть выведена информация о номерах, цены на которые ниже указанного значения) Для удобства работы реализовать меню с соответствующими пунктами.

Тема Рекурсия

Вариант 1

Вычислить значение n-го члена ряда Фибоначчи: f(0)=0, f(1)=1, f(n)=f(n-1)+f(n-2).

Вариант 2

Вычислить произведение двух целых положительных чисел (используя суммирование).

Вариант 3

Подсчитать, сколько раз встречается некоторое слово в строке. Строка и слово должны вводиться с клавиатуры. Для разделения строки на слова использовать стандартный предикат fronttoken (String, Lexeme, StringRest), позволяющий разделить строку String на первое слово Lexeme и остаток строки StringRest.

Вариант 4

Поменять порядок следования букв в слове на противоположный. Для разделения строки на символы использовать стандартный предикат frontchar (String, Char, StringRest), позволяющий разделять строку String на первый символ Char и остаток строки StringRest.

Вариант 5

Вычислить сумму ряда целых нечетных чисел от 1 до n.

Вариант 6

Поменять порядок следования слов в предложении на противоположный. Для разделения строки на слова использовать стандартный предикат fronttoken (String, Lexeme, StringRest), позволяющий разделить строку String на первое слово Lexeme и остаток строки StringRest..

Вариант 7

Вычислить сумму ряда целых четных чисел от 2 до n.

Вариант 8

Организовать ввод целых положительных чисел и их суммирование до тех пор, пока сумма не превысит некоторого порогового значения. Введенные отрицательные целые числа суммироваться не должны.

Вариант 9

Организовать ввод букв и их соединение в строку до тех пор, не будет введен символ #. Для присоединения символа к строке использовать стандартный предикат frontchar (String, Char, StringRest), позволяющий присоединять символ Char к строке StringRest и получать строку String.

Вариант 10

Подсчитать, сколько раз встречается некоторая буква в строке. Строка и буква должны вводиться с клавиатуры. Для разделения строки на символы использовать стандартный предикат frontchar (String, Char, StringRest), позволяющий разделять строку String на первый символ Char и остаток строки StringRest.

Тема Рекурсивные структуры данных (списки)

Вариант 1

Написать программу для получения значения n-го элемента списка. Например: в списке [three, one, two] второй элемент равен one.

Вариант 2

Написать программу для удаления из списка всех элементов, равных 0. Например: список [1, 0, 2, 0, 0, 3] преобразуется в список [1, 2, 3].

Вариант 3

Написать программу для циклического сдвига списка вправо на заданное число элементов. Например: список [6, 5, 4, 3, 2, 1], циклически сдвинутый вправо на 2 элемента, преобразуется в список [2, 1, 6, 5, 4, 3].

Вариант 4

Написать программу для удаления из списка 2-ого, 4-ого и т.д. элементов. Например: список [6, 5, 4, 3, 2, 1] преобразуется в список [6, 4, 2].

Вариант 5

Написать программу для замены в списке всех элементы, равные 0, на -1. Например: список [1, 0, 0] преобразуется в список [1, -1, -1].

Вариант 6

Написать программу для перевода списка арабских чисел (от 1 до 10) в список римских чисел. Например: список [1, 2, 3] преобразуется в список [“I”, “II”, “III”].

Вариант 7

Написать программу для подсчета количества определенных элементов в списке. Например: в списке [1, 2, 1, 3, 1] три единицы.

Вариант 8

Написать программу для подсчета количества элементов списка без какого-либо указываемого элемента. Например: в списке [1, 2, 1, 3, 1] два элемента без учета единиц.

Вариант 9

Написать программу для подсчета количества элементов списка, значения которых лежат в определенном диапазоне. Например: в списке [10, 20, 10, 30, 15] два элемента, значения которых больше 10 и меньше 30.

Вариант 10

Написать программу для реверса списка. Например: список [1, 2, 3] преобразуется в список [3, 2, 1].

Тема Рекурсивные структуры данных (деревья)

Вариант 1

Написать программу для проверки упорядоченности бинарного дерева.

Вариант 2

Вывести бинарное дерево на экран в виде дерева.

Вариант 3

Написать программу для вычисления глубины бинарного дерева (глубина пустого дерева равна 0, глубина одноузлового дерева равна 1).

Вариант 4

Написать программу для подсчета количества листьевых вершин дерева, значения которых лежат в определенном диапазоне.

Вариант 5
Написать программу для преобразования дерева в список.
Вариант 6
Написать программу для нахождения среднего арифметического отрицательных узлов дерева.
Вариант 7
Написать программу для подсчета количества вершин бинарного дерева, значения которых не равны 0.
Вариант 8
Написать программу для нахождения среднего арифметического положительных узлов дерева.
Вариант 9
Написать программу для подсчета количества вершин бинарного дерева, значения которых равны 0.
Вариант 10
Написать программу для нахождения среднего арифметического листьевых вершин бинарного дерева.

19.3.9 Комплект разноуровневых заданий

1 Составление глоссария и кластера основных терминов раздела (нескольких разделов) дисциплины (реконструктивный уровень)

2 Составление сравнительных, концептуальных таблиц по заданной теме (творческий уровень)

3 Составление, коррекция синквейнов и денотатных графов с основными понятиями (творческий уровень)

4 Составление аннотированного перечня источников сети Интернет (реконструктивный уровень)

5 Написание рецензий на готовые рефераты по разделам дисциплины, скачанные с различных сайтов (творческий уровень)

6 Составление таблицы толстых и тонких вопросов по разделам дисциплины (реконструктивный уровень)

7 Составление вопросов к ромашке Блума (таксономия целей) к разделам дисциплины (творческий уровень)

19.4. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности, характеризующих этапы формирования компетенций

Оценка знаний, умений и навыков, характеризующая этапы формирования компетенций в рамках изучения дисциплины осуществляется в ходе текущей и промежуточной аттестаций.

Текущий контроль успеваемости проводится в соответствии с Положением о текущей аттестации обучающихся по программам высшего образования Воронежского государственного университета. Текущий контроль успеваемости проводится в формах*: фронтальных опросов, защиты индивидуальных заданий, контрольных и лабораторных работ, выполнения рефератов, тестирования.* Критерии оценивания приведены выше.

Промежуточная аттестация проводится в соответствии с Положением о промежуточной аттестации обучающихся по программам высшего образования.

Контрольно-измерительные материалы промежуточной аттестации включают в себя теоретические вопросы, позволяющие оценить уровень полученных знаний и практическое задание, позволяющее оценить степень сформированности умений и навыков.

При оценивании используются качественные шкалы оценок. Критерии оценивания приведены выше.