МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«Национальный исследовательский ядерный университет «МИФИ»
Обнинский институт атомной энергетики –
филиал федерального государственного автономного образовательного учреждения высшего образования
«Национальный исследовательский ядерный университет «МИФИ»
(ИАТЭ НИЯУ МИФИ)

Одобрено УМС ИАТЭ НИЯУ МИФИ,
Протокол №2-8/2021 От 30.08.2021

РАБОЧАЯ ПРОГРАММА УЧЕБНОЙ ДИСЦИПЛИНЫ

ЛОГИЧЕСКОЕ ПРОГРАММИРОВАНИЕ
(ЛПрг)
(Наименование дисциплины)

09.03.01 - Информатика и вычислительная техника
(Код (шифр), наименование направления подготовки (специальности) ФГОС)

Профиль: **«Вычислительные машины, комплексы, системы и сети»**

(Профиль направления)

Название программы бакалавриата

бакалавр
(Квалификация (степень) выпускника)

очная
Форма обучения (очная, очно-заочная (вечерняя), заочная)

г. Обнинск 2021 г.

Программа составлена в соответствии с требованиями образовательного стандарта высшего образования национального исследовательского ядерного университета «МИФИ» по направлению подготовки 09.03.01 – Информационные системы и технологии (уровень бакалавриат).

Автор(ы)

_____ А.В. Васяшин, ст. преподаватель каф. АСУ

Рецензент(ы)

Программа рассмотрена на заседании отделения интеллектуальных кибернетических систем (О)
(протокол № 5/7 от «30» июля 2021 г.)

Руководитель образовательной программы
09.03.01 Информатика и вычислительная техника
_____ С.О. Старков
«30» июля 2021 г.

1. Перечень планируемых результатов обучения по дисциплине, соотнесенных с планируемыми результатами освоения образовательной программы

В результате освоения ООП бакалавриата обучающийся должен овладеть следующими результатами обучения по дисциплине:

Коды компетенций	Результаты освоения ООП *Содержание компетенций*	Перечень планируемых результатов обучения по дисциплине
ПК-3	Способен разрабатывать модели и компоненты аппаратно-программных комплексов и баз данных, используя современные инструментальные средства и технологии	Знать: основы и принципы логического программирования, возможности использования логического программирования в решении задач искусственного интеллекта. Уметь: разрабатывать программы и продукционные системы на языке «Пролог». Владеть: основными приемами и методами разработки программ на языке «Пролог», способами организации дедуктивных баз данных и реализации символических вычислений в языках логического программирования.

2. Место дисциплины в структуре ООП бакалавриата

Дисциплина реализуется в рамках вариативной части.

Для освоения дисциплины необходимы компетенции, сформированные в рамках изучения следующих дисциплин: «Дискретная математика».

Дисциплина изучается на3 курсе в 5 семестре.

3. Объем дисциплины в зачетных единицах с указанием количества академических часов, выделенных на контактную работу обучающихся с преподавателем (по видам занятий) и на самостоятельную работу обучающихся

Общая трудоемкость (объем) дисциплины составляет 4 зачетных единицы (з.е.), 144 академических часов.

3.1. Объём дисциплины по видам учебных занятий (в часах)

Объем дисциплины	Всего часов
	Очная форма обучения
Общая трудоемкость дисциплины	144
Контактная работа обучающихся с преподавателем (по видам учебных занятий) (всего)	68
Аудиторная работа (всего):	48
лекции	16
семинары, практические занятия	16
лабораторные работы	16
Внеаудиторная работа (всего):	-
индивидуальная работа обучающихся с преподавателем:	20
курсовое проектирование	
групповая, индивидуальная консультация и иные виды учебной деятельности, предусматривающие групповую или индивидуальную работу обучающихся с преподавателем	
творческая работа (эссе)	

Самостоятельная работа обучающихся (всего)	76
Вид промежуточной аттестации обучающегося (зачет/экзамен(часы))	Зачет с оценкой

4. Содержание дисциплины, структурированное по темам (разделам) с указанием отведенного на них количества академических часов и видов учебных занятий

4.1. Разделы дисциплины и трудоемкость по видам учебных занятий (в академических часах)

№ п/п	Наименование раздела /темы дисциплины	Общая трудоёмкость всего *(в часах)*	Виды учебных занятий, включая самостоятельную работу обучающихся и трудоемкость *(в часах)*				Формы текущего контроля успеваемости
			Аудиторные учебные занятия*			СРО	
			Лек	Сем/Пр	Лаб		
1.	Введение в логическое программирование						Лабораторные работы 0, 1, Контрольная работа 1
1.1.	Парадигма логического программирования.	3	1		0	2	
1.2.	Запросы	4	1		1	2	
1.3	Термы «Пролога»	3	1		0	2	
1.4	Факты и правила. Структура программы.	4	1		1	2	
1.5	Унификация	4	1		1	2	
1.6	Арифметика языка.	3	1		0	2	
1.7	Поиск с возвратом.	4	1		1	2	
2.	Списки						Лабораторные работы 1, 2, Контрольная работа 1
2.1.	Введение в использование списков	5	1		2	2	
2.2.	Исходящая рекурсия.	10	2		4	4	
2.3	Технология разработки рекурсивных процедур.	10	2		4	4	
2.4	Входящая рекурсия.	5	1		2	2	
3.	Структуры						Лабораторные работы 3, 4, Контрольная работа 1
3.1.	Введение в использование структур	5	1		2	2	
3.2.	Бинарные структуры.	7	1		4	2	
3.3	Произвольные структуры.	12	2		6	4	
3.4	Операторы языка.	6	2		0	4	
4.	Интерпретатор и способы управления его работой.						Лабораторная работа 5,

							Контрольная работа 2
4.1.	Обработка правила.	3	1		0	2	
4.2.	Сокращение пространства поиска ответов.	3	1		0	2	
4.3	Вынуждаемый возврат.	11	1		6	4	
4.4	Счётчики и предикаты работы с ними.	3	1		0	2	
4.5	Работа с базой знаний	3	1		0	2	
5.	Организация термов в базы данных						Лабораторная работа 5, Контрольная работа 2
5.1.	Базы данных под ключом.	6	2		0	4	
5.2.	Предикаты для БД под ключом.	6	2		0	4	
6.	Пролог – язык искусственного интеллекта.						Контрольная работа 2
6.1	DCG-расширение языка.	8	2		0	6	
6.2	Экспертные системы продукционного типа и их разработка.	8	2		0	6	
6.3	Базы данных и знаний. Дедуктивные базы данных.	8	2		0	6	

*Прим.: Лек – лекции, Сем/Пр – семинары, практические занятия, Лаб – лабораторные занятия, СРО – самостоятельная работа обучающихся

4.2. Содержание дисциплины, структурированное по разделам (темам)

4.2.1. Лекционный курс

№	Наименование раздела /темы дисциплины	Содержание
1.	Введение в логическое программирование	
1.1.	Парадигма логического программирования.	Идея создания языка программирования на основе логики предикатов первого порядка (работы Р. Ковальски). Первая реализация языка во Франции. Абстрактная машина Уоррена. Эдинбургский стандарт языка. Обзор существующих версий языка «Пролог».
1.2.	Запросы	Элементарная программа на языке «Пролог» и запросы к ней. Простые и составные запросы. Семантика запросов. Программа – база знаний, а интерпретатор – машина вывода.
1.3	Термы «Пролога»	Классификация термов. Простые и составные термы. Правила написания.
1.4	Факты и правила. Структура программы.	Программа как набор фраз. Два типа фраз. Заголовок и тело правила. Понятие процедуры (предиката). Детерминированные и недетерминированные предикаты.
1.5	Унификация	Свободные и связанные переменные. Понятие и определение унификации. Оператор унификации.
1.6	Арифметика языка.	Основные операторы арифметики. Оператор «`is`». Отсутствие деструктивного присвоения. Реализация расчета факториала.

1.7	Поиск с возвратом.	Неудача согласования. Поиск с возвратом. Исчерпывающий поиск ответа.
2.	Списки	
2.1.	Введение в использование списков	Определение. Обобщенное представление и конструктор списка. Унификация списков. Встроенные предикаты: `sort`, `length` и `findall`.
2.2.	Исходящая рекурсия.	Исходящая рекурсия – основной приём программирования. Рекурсивные фразы и фразы останова рекурсии. Простые примеры использования: подсчет элементов списка, сумма элементов числового списка.
2.3	Технология разработки рекурсивных процедур	Три этапа разработки: формулировка алгоритма в рекурсивной форме, кодирование, «ручная трассировка» для простых запросов. Разработка на примере предикатов `append` и `member`.
2.4	Входящая рекурсия.	Входящая рекурсия – приём организации рекурсии. Возврат ответа во входящей рекурсии. Пример использования для инверсии списка.
3.	Структуры	
3.1.	Введение в использование структур	Определение структуры. Имя, арность и оператор «`=..`». Предикаты: `arg`, `functor`. Структуры и понятия теории графов (деревья, корень, листья, предок, потомки, поддерево, частичное поддерево, глубина дерева).
3.2.	Бинарные структуры.	Технология обработки бинарных структур. Рекурсивная обработка структур и правило останова рекурсии. Предикат `atomic`.
3.3	Произвольные структуры.	Способы обработки произвольных структур. Деление на поддерево и частичное поддерево (первый способ). Обработка списка поддеревьев: косвенная рекурсия (второй способ) и использование предиката `member` (третий способ).
3.4	Операторы языка.	Работа с операторами, предикаты: `current_op` и `op`. Старшинство и ассоциативность операции. Введение собственных операторов и расширение синтаксиса языка на примере операций над множествами.
4.	Интерпретатор и способы управления его работой.	
4.1.	Обработка правила.	Порядок обработки подцелей правила. Предикаты: `ifthen`, `ifthenelse`, `case`, `true` и `fail`.
4.2.	Сокращение пространства поиска ответов.	Сокращение пространства поиска ответов. Предикат «сократить». Отрезок («[!» и «!]»).
4.3	Вынуждаемый возврат.	Прием программирования – вынуждаемый возврат. Предикат `repeat`. Имитация циклической обработки. Работа с файлами.
4.4	Счётчики и предикаты работы с ними.	Счетчики. Организация целочисленных глобальных переменных. Предикаты работы со счетчиками. Использование счетчиков при вынуждаемом возврате.
4.5	Работа с базой знаний.	Работа с базой знаний. Изменение базы знаний. Организация «глобальных переменных». Предикаты `assert` и `retract`.
5.	Организация термов в базы данных.	
5.1.	Базы данных под ключом.	База данных – цепочка термов под ключом. Ссылочные номера термов.
5.2.	Предикаты для БД под ключом.	Предикаты: `recordz`, `recorda`, `recorded`, `instance`, `erase`, `eraseall`. Циклическая обработка цепочек термов.

6.	Пролог – язык искусственного интеллекта.	
6.1	DCG-расширение языка.	Обработка текстов на естественном языке. DCG-расширение языка. Контекстно-свободная и контекстно-чувствительная обработка.
6.2	Экспертные системы продукционного типа и их разработка.	Системы продукций. Виды продукций. Пролог - продукционный язык. Введение в экспертные системы. Обобщенная структура экспертной системы. Реализация продукционных систем.
6.3	Базы данных и знаний. Дедуктивные базы данных.	Дедуктивные базы данных. Сравнение дедуктивных и реляционных БД. Реализация дедуктивных БД в виде надстройки в реляционных СУБД. Datalog.

4.2.2. Практические/семинарские занятия

Не предусмотрены.

4.2.3. Лабораторные занятия

№	**Наименование раздела /темы дисциплины**	**Название лабораторной работы**
1.	Введение в логическое программирование	
1.1.	Парадигма логического программирования.	Освоение интерпретатора Пролога. (ЛР №0).
1.2.	Запросы	
1.3	Термы «Пролога»	
1.4	Факты и правила. Структура программы.	
1.5	Унификация	
1.6	Арифметика языка.	
1.7	Поиск с возвратом.	
2.	Списки	
2.1.	Введение в использование списков	ЛР по курсу: - разработка детерминированной процедуры обработки списков (ЛР №1); - разработка недетерминированной процедуры обработки списков (ЛР №2).
2.2.	Исходящая рекурсия.	
2.3	Технология разработки рекурсивных процедур	
2.4	Входящая рекурсия.	
3.	Структуры	
3.1.	Введение в использование структур	Разработка процедуры обработки бинарной структуры (ЛР №3).
3.2.	Бинарные структуры.	
3.3	Произвольные структуры.	Разработка процедуры обработки произвольной структуры (ЛР №4)
3.4	Операторы языка.	
4.	Интерпретатор и способы управления его работой.	
4.1.	Обработка правила.	Разработка процедуры обработки данных методом вынуждаемого возврата (ЛР №5).
4.2.	Сокращение пространства поиска ответов.	
4.3	Вынуждаемый возврат.	
4.4	Счётчики и предикаты работы с ними.	
4.5	Работа с базой знаний.	
5.	Организация термов в базы данных.	

5.1.	Базы данных под ключом.	См. ЛР предыдущего раздела.
5.2.	Предикаты для БД под ключом.	
6.	Пролог – язык искусственного интеллекта.	
6.1	DCG-расширение языка.	Не предусмотрено.
6.2	Экспертные системы продукционного типа и их разработка.	
6.3	Базы данных и знаний. Дедуктивные базы данных.	

5. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине

В качестве учебно-методических материалов используется рекомендованная литература.

6. Фонд оценочных средств для проведения промежуточной аттестации обучающихся по дисциплине

6.1. Паспорт фонда оценочных средств по дисциплине

№ п/п	Контролируемые разделы (темы) дисциплины (результаты по разделам)	Код контролируемой компетенции (или её части) / и ее формулировка	Наименование оценочного средства
Текущий контроль			
1.	Введение в логическое программирование	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторные работы 0, 1, Контрольная работа 1
2.	Списки	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторные работы 1, 2, Контрольная работа 1
3.	Структуры	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторные работы 3, 4, Контрольная работа 1
4.	Интерпретатор и способы управления его работой	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторная работа 5, Контрольная работа 2
5.	Организация термов в базы данных	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Лабораторная работа 5, Контрольная работа 2
6.	Пролог – язык искусственного интеллекта	ПК-3: Способность осваивать методики использования программных средств	Контрольная работа 2

		для решения практических задач	
Промежуточный контроль			
	Зачет по всем темам	ПК-3: Способность осваивать методики использования программных средств для решения практических задач	Отчет по ЛР и две задачи

6.2. Типовые контрольные задания или иные материалы

6.2.1. Зачет

Зачет (промежуточный контроль) выставляется по результатам решения двух задач (см. контрольные работы 1 и 2) и отчет по лабораторным работам. Итоговая оценка включает в себя оценки за две контрольные и каждую из пяти лабораторных работ (текущий контроль, суммарно 60%) и оценку за промежуточный контроль (40%).

6.2.2 Типовые задания к лабораторным работам 1-5

См. учебное пособие [1, доп. литература].

6.2.3 Контрольная работа №1

Задача №1

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом влево на случайное число позиций.

Задача №2

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного удалением случайного числа элементов исходного списка. Вероятность вхождения элемента исходного списка в результат, должна быть равна вероятности его отсутствия в результирующем списке.

Задача №3

Разработать предикат, который по двум исходным, упорядоченным по возрастанию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением исходных списков, который также должен быть упорядочен по возрастанию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №4

Разработать предикат, который по двум исходным спискам (первый и второй аргументы), определяет, являются ли списки одинаковыми (с точностью до перестановки элементов). Предполагается, что каждый из исходных списков может иметь повторяющиеся элементы.

Задача №5

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий листья исходного дерева.

Задача №6

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №7

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, удалением одного из его

поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №8

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет являются ли они эквивалентными.

Задача №9

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет является ли первое из них поддеревом второго.

Задача №10

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки поддеревьев. Перестановка производится только тогда, когда правым поддеревом является лист дерева, а левое поддерево листом не является.

Задача №11

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет является ли первое из них частичным поддеревом второго.

Задача №12

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий наименования вершин лежащих на пути от корня дерева до листа. В ходе последовательного отказа от ответа предикат должен возвращать все такие пути.

Задача №13

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (третий аргумент), содержащий наименования вершин лежащих на глубине, заданной вторым аргументом.

Задача №14

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины всех путей от корня дерева до его листьев.

Задача №15

Разработать предикат, который по исходному, произвольному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №16

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом вправо на случайное число позиций.

Задача №17

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного дублированием каждого элемента исходного списка.

Задача №18

Разработать предикат, который по двум исходным, упорядоченным по убыванию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением

исходных списков, который также должен быть упорядочен по убыванию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №19

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий такие узлы, у которых среди дочерних узлов есть листья исходного дерева.

Задача №20

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, дублированием одного из его поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №21

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет являются ли они эквивалентными.

Задача №22

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки листьев (если оба дочерних узла дерева - листья).

Задача №23

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины путей от корня дерева до его листьев.

Задача №24

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом влево на случайное число позиций.

Задача №25

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного удалением случайного числа элементов исходного списка. Вероятность вхождения элемента исходного списка в результат, должна быть равна вероятности его отсутствия в результирующем списке.

Задача №26

Разработать предикат, который по двум исходным, упорядоченным по возрастанию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением исходных списков, который также должен быть упорядочен по возрастанию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №27

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий листья исходного дерева.

Задача №28

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №29

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, удалением одного из его поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №30

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет являются ли они эквивалентными.

Задача №31

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет является ли первое из них поддеревом второго.

Задача №32

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки поддеревьев. Перестановка производится только тогда, когда правым поддеревом является лист дерева, а левое поддерево листом не является.

Задача №33

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет является ли первое из них частичным поддеревом второго.

Задача №34

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий наименования вершин лежащих на пути от корня дерева до листа. В ходе последовательного отказа от ответа предикат должен возвращать все такие пути.

Задача №35

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (третий аргумент), содержащий наименования вершин лежащих на глубине, заданной вторым аргументом.

Задача №36

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины всех путей от корня дерева до его листьев.

Задача №37

Разработать предикат, который по исходному, произвольному дереву (первый аргумент), возвращает поддерево (второй аргумент) исходного дерева. При отказе от ответа, предикат должен возвращать другое поддерево и т.д. пока не будут возвращены все поддеревья.

Задача №38

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного циклическим сдвигом вправо на случайное число позиций.

Задача №39

Разработать предикат, который по исходному списку (первый аргумент), возвращает список (второй аргумент), полученный из исходного дублированием каждого элемента исходного списка.

Задача №40

Разработать предикат, который по двум исходным, упорядоченным по убыванию спискам (первый и второй аргументы), возвращает список (третий аргумент), являющийся объединением исходных списков, который также должен быть упорядочен по убыванию. (Примечание: запрещается использовать встроенный предикат **sort**)

Задача №41

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает список (второй аргумент), содержащий такие узлы, у которых среди дочерних узлов есть листья исходного дерева.

Задача №42

Разработать предикат, который по исходному бинарному дереву (первый аргумент), возвращает дерево (второй аргумент), получающееся из исходного дерева, дублированием одного из его поддеревьев. При отказе от ответа, предикат должен возвращать другое дерево и т.д. пока не будут возвращены все деревья, которые можно получить из исходного.

Задача №43

Разработать предикат, который по двум исходным бинарным деревьям (первый и второй аргументы), определяет являются ли они эквивалентными.

Задача №44

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит дерево (второй аргумент), путем перестановки листьев (если оба дочерних узла дерева - листья).

Задача №45

Разработать предикат, который для исходного бинарного дерева (первый аргумент), строит список (второй аргумент), содержащий длины путей от корня дерева до его листьев.

6.2.4 Контрольная работа №2

В данной контрольной работе проверяются знания и навыки работы с термами базы данных и использования одного из основных приёмов программирования - «вынуждаемый возврат». В каждом задании необходимо:

1. Объяснить, что делает предложенный предикат;
2. Указать, какие аргументы предиката должны быть конкретизированы при запросе к нему;
3. Привести все возможные варианты (по конкретизации аргументов) примеров запросов к предикату и раскрыть результаты таких запросов;
4. Объяснить работу интерпретатора по выполнению одного из возможных запросов;
5. Если в качестве подцели в теле одной или нескольких фраз предиката выступает предикат «сократить», пояснить смысл его использования.

Варианты заданий

Вариант №1
```
f(Key):-
        recorded(Key,T,U),
        recorda(Key,T,U),
        not(nref(U,_)),
        !.
```

Вариант №2
```
g(Key1,Key2):-
        recorded(Key1,T,_),
        not(recorded(Key2,T,_)),
        !.
```

Вариант №3
```
g(Key):-
        recorded(Key,T,R),
        nref(R,N),
        instance(N,F),
        F < T.
```

Вариант №4
```
f(Key1,Key2,T):-
        recorded(Key1,T,_),
        recorded(Key2,T,_).
```

Вариант №5
```
f_replace(Key):-
        recorded(Key,Term,R),
        Term=..L,
        replace(R,L),
        fail.
```

Вариант №6
```
f_assert(Key):-
        recorded(Key,Term,_),
        assert(Term),
        fail.
f_assert(Key):-
        eraseall(Key).
```

Вариант №7
```
c_assert(Key):-
        recorded(Key,Term,_),
        (assert(Term) ; retract(Term)).
```

Вариант №8
```
d(Key):-
        recorded(Key,Term,R1),
        recorded(Key,Term,R2),
        R1 \== R2,
        !.
```

Вариант №9
```
g(Key1,Key2):-
        recorded(Key1,T,R),
        not(recorded(Key2,T,_)),
        recordz(Key2,T,_),
        fail.
```

Вариант №10
```
r(Key,Term):-
        recorded(Key,Term,R),
        erase(R),
        !.
dr(Key,Term):-
        repeat,
        not(r(Key,Term)),
        !,
        recordz(Key,Term,_).
```

Вариант №11
```
c(Key,Term,N):-
        findall(1,
                recorded(Key,Term,_),
                L),
        length(L,N).
```

Вариант №12
```
d(Key):-
        ctr_set(0,1),
        recorded(Key,T,R),
        ctr_inc(0,N),
        S=..[N,T],
        replace(R,S),
        fail.
```

Вариант №13
```
g(Key,Term):-
        recorded(Key,Term,R),
        nref(R,N),
        instance(N,Term).
```

Вариант №14
```
a(_):-
        ctr_set(0,0),
        recorded(f,_,R),
        ctr_inc(0,_),
        fail.
a(X):-
        ctr_is(0,X).
```

Вариант №15 h(Key):- recorded(Key,T,U), nref(U,R), instance(R,G), recorda(Key,G,_), erase(R), fail.	Вариант №16 g(Key,Ref):- recordz(Key,$$,R), pref(R,Ref), erase(Ref).
Вариант №17 n(Key,Term,N):- ctr_set(0,1), repeat, ctr_inc(0,X), nth_ref(Key,X,R), instance(R,T), T = Term, N = X.	Вариант №18 g(Key):- recorded(Key,T,U), nref(U,R), instance(R,G), recorda(Key,G,_), fail.
Вариант №19 n_recorded(Key,Term,Ref):- repeat, recorded(Key,Term,Ref).	Вариант №20 b(X,Y):- ctr_set(0,X), repeat, ctr_is(0,N), Y is N*(N+1), ctr_set(0,Y).
Вариант №21 n_recordz(Key,Term.Ref):- recordz(Key,Term,Ref), (true; (erase(Ref), fail)).	Вариант №22 f(Key):- recorded(Key,T,Ref), !, repeat, ifthenelse(nref(Ref,N), (instance(N,Z), erase(N), recorda(Key,Z,_), fail), true), !.
Вариант №23 r(Key):- recorded(Key,T,R), nref(R,N), erase(N), fail.	Вариант №24 r(Key):- recorded(key,T,R), pref(R,N), erase(N), fail.

Вариант №25 r(Key):- recorded(key,T,R), pref(R,N1), pref(N1,N), erase(N), fail.	Вариант №26 c(Key):- recorded(Key,T,R), erase(R), recordz(Key,T,_), !. dc(Key):- repeat, c(Key).
Вариант №27 f(Key):- recorded(Key,T,R), pref(R,P), instance(P,V), ifthen(V > T, erase(P)), fail.	Вариант №28 g(Key1,Key2):- recorded(Key1,_,Ref), nref(Ref,N), instance(N,T), erase(N), recordz(Key2,T,_), fail.
Вариант №29 h(Key):- recorded(key,T,Ref), nref(Ref,N), instance(N,V), replace(Ref,(T,V)), erase(N), fail.	

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

Форма аттестации	Наименование оценочного средства	Баллы
Зачет (100 баллов)	Лабораторные работы	48
	Контрольная работа № 1	6
	Контрольная работа № 2	6
	Итоговый контроль	40

7. Перечень основной и дополнительной учебной литературы, необходимой для освоения дисциплины

а) основная учебная литература

1. Волчёнков, Н.Г. Логическое программирование. Язык Пролог [Электронный ресурс]: тексты лекций / Н. Г. Волчёнков. - 2-е изд., испр. и доп. - Москва : НИЯУ МИФИ, 2015. – ISBN 978-5-7262-2091-8 (электронный ресурс, доступен в Электронно-библиотечной системе Центра информационно-библиотечного обеспечения учебно-научной деятельности НИЯУ МИФИ, режим доступа – http://library.mephi.ru/Data-Irbis/book-mephi/Volchenkov_Logicheskoe_programmirovanie_yazyk_prolog_2015.pdf)
2. Орлов С. Теория и практика языков программирования. Учебник для вузов. СПб: Питер, 2013 г. – 688 стр. (24 экз)
3. Сошников Д.В. Логическое программирование. 2011 г. (электронный курс, режим доступа – http://www.intuit.ru/studies/courses/558/414/info)
4. Ефимова Е.А. Основы программирования на языке Visual Prolog. 2014 г. (электронный курс, режим доступа – http://www.intuit.ru/studies/courses/12333/1180/info)

5. Ефимова Е.А. Разработка приложений на языке Visual Prolog. 2015 г. (электронный курс, режим доступа – http://www.intuit.ru/studies/courses/3507/749/info)

б) дополнительная учебная литература

1. Васяшин А.В. Сборник задач по курсу «Логическое программирование»: Учебное пособие. – Обнинск: ИАТЭ, 2009. – 68 с. (50 экз.)
2. Бессмертный И.А. Искусственный интеллект: Учебное пособие. – СПб: СПбГУ ИТМО, 2010. – 132 с. (2 экз. электронный ресурс, режим доступа – http://window.edu.ru/resource/274/69274)
3. Подольский В.Е. Методы искусственного интеллекта для синтеза проектных решений: Учебное пособие. – Тамбов: Издательство ТГТУ, 2010. – 80 с. (электронный ресурс, режим доступа – http://window.edu.ru/resource/173/73173)
4. Солдатова О.П., Лезина И.В. Логическое программирование на языке Visual Prolog: учебное пособие. – Самара: СНЦ РАН, 2010. – 81 с. (электронный ресурс, режим доступа – http://window.edu.ru/resource/698/69698)
5. Ездаков А.Л. Функциональное и логическое программирование: Учебное пособие. – М.: БИНОМ. Лаборатория знаний, 2009. – 119 с. (электронный ресурс, режим доступа – http://window.edu.ru/resource/300/65300)
6. Братко И. Программирование на языке Пролог для искусственного интеллекта: Пер. с англ. – М.: Мир, 1990. – 560 с. (12 экз.)
7. Стерлинг Л., Шапиро Э. Искусство программирования на языке Пролог: Пер. с англ. – М.: Мир, 1990. – 235 с. (24 экз.)
8. Бураков М.В. Язык логического программирования ПРОЛОГ: Методические указания к выполнению лабораторных работ. – СПб.: ГУАП, 2003. – 37 с. (электронный ресурс, режим доступа – http://window.edu.ru/resource/733/44733)
9. Дубинин В.Н., Зинкин С.А. Языки логического программирования в проектировании вычислительных систем и сетей: Учеб. пособие. – Пенза: Изд-во Пенз. гос. техн. ун-та, 1997. – 88 с. (электронный ресурс, режим доступа – http://window.edu.ru/resource/984/23984)
10. Новицкая Ю.В. Основы логического и функционального программирования: Учебное пособие. – Новосибирск: НГТУ, 2006. – 60 с. (электронный ресурс, режим доступа – http://window.edu.ru/resource/286/29286)
11. Чанышев О.Г. ПРОграммирование в ЛОГике: Учебное пособие. – Омск: Изд-во ОмГУ, 2004. – 64 с. (электронный ресурс, режим доступа – http://window.edu.ru/resource/783/27783)

8. Перечень ресурсов информационно-телекоммуникационной сети «Интернет» (далее - сеть «Интернет»), необходимых для освоения дисциплины

1. Материалы открытой энциклопедии Wikipedia // Корневая URL: http://ru.wikipedia.org/wiki/Логическое программирование
2. Ресурсы портала «Единое окно доступа к образовательным ресурсам» / Раздел «Программирование» // URL: http://window.edu.ru/catalog/?p_rubr=2.2.75.6.14
3. Ресурсы электронно-библиотечной системы Центра информационно-библиотечного обеспечения учебно-научной деятельности НИЯУ МИФИ // URL: www.library.mephi.ru (по подписке)
4. Ресурсы научной электронной библиотеки elibrary.ru // URL: www.elibrary.ru (по подписке)
5. Ресурсы электронно-библиотечной системы издательства «Лань» // URL: www.e.lanbook.com (по подписке)
6. Ресурсы электронно-библиотечной системы образовательных и просветительских изданий // URL: www.iqlib.ru (по подписке)

9. Методические указания для обучающихся по освоению дисциплины

Вид учебного занятия	Организация деятельности студента
Лекция	Написание конспекта лекций: кратко, схематично, последовательно фиксировать основные положения, выводы, формулировки, обобщения; помечать важные мысли, выделять ключевые слова, термины. Проверка терминов, понятий с помощью энциклопедий, словарей, справочников с выписыванием толкований в тетрадь. Обозначить вопросы, термины, материал, который вызывает трудности, пометить и попытаться найти ответ в рекомендуемой литературе. Если самостоятельно не удается разобраться в материале, необходимо сформулировать вопрос и задать преподавателю на консультации, на практическом занятии и лабораторной работе. Уделить внимание следующим понятиям: терм, свободная и связанная переменная, унификация, конкретизация, входящая и исходящая рекурсии, поиск с возвратом, отсечение, вынуждаемый возврат.
Контрольная работа / индивидуальные задания	Попрактиковаться в решении задач.
Практикум / лабораторная работа	При выполнении лабораторных работ необходимо ориентироваться на конспекты лекций, рекомендуемую литературу и др.
Подготовка к экзамену (зачету)	Попрактиковаться в решении задач.

10. Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине, включая перечень программного обеспечения и информационных справочных систем (при необходимости)

1. Интерпретатор Пролога (рекомендуется использовать Arity/Prolog или SWI-Prolog).

2. Система помощи и программная документация от производителя интерпретатора.

3. Набор примеров с сайта производителя.

11. Описание материально-технической базы, необходимой для осуществления образовательного процесса по дисциплине

Класс персональных ЭВМ, видеопроектор, текстовый редактор Microsoft Word для подготовки отчетов.

12. Иные сведения и (или) материалы

12.1. Перечень образовательных технологий, используемых при осуществлении образовательного процесса по дисциплине

Часов в интерактивной форме – 16.

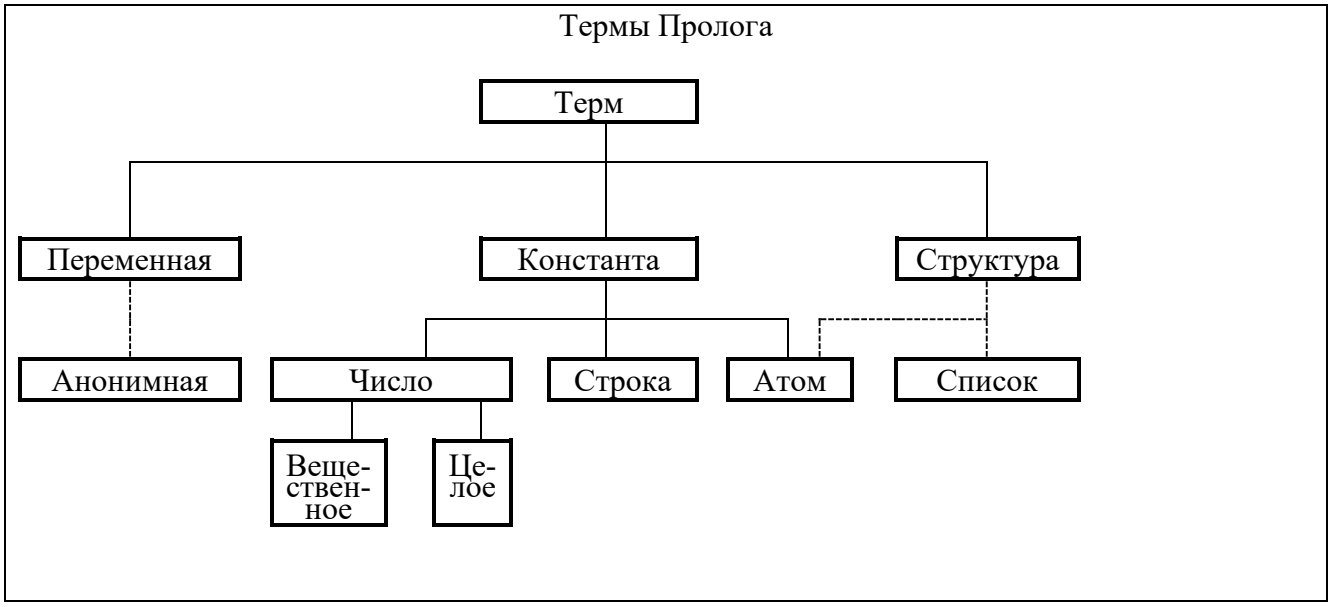
В ходе выполнения заданий на лабораторные работы производится обсуждение возможных вариантов решения. Под вариантом решения понимается выбранный алгоритм, необходимые встроенные предикаты и т.п. После решения (выполнения работы) обсуждается эффективность полученного решения.

12.2. Формы организации самостоятельной работы обучающихся (темы, выносимые для самостоятельного изучения; вопросы для самоконтроля; типовые задания для самопроверки

Основным критерием для самопроверки является умение решать задачи средствами логического программирования, поэтому в качестве самопроверки следует использовать задачи контрольных и лабораторных работ. Правильность решения проверяется учащимся непосредственно в интерпретаторе Пролога.

12.3. Краткий терминологический словарь

- **Аргументность** (или **арность**, arity) - количество аргументов структуры или предиката;
- **Атом** (atom) - это нечисловая константа (например: hello, ‘New York’, ‘май’);
- **Детерминированный** (determinate). Процедура является детерминированной, если она успешно выполняется лишь один раз. Составной запрос (или тело правила) будет детерминированным, если после его выполнения интерпретатор окажется не в состоянии осуществить поиск с возвратом ни для какой подцели этого запроса;
- **Запрос** (или цель, query or goal). С позиций декларативной семантики, запрос - это вопрос, который пользователь задает Пролог - системе о том, соблюдается ли конкретная реализация отношения. С позиций процедурной семантики, запрос - это вызов процедуры;
- **Конкретизация** (instantiation). Переменная конкретизируется, когда при выполнении запроса она получает в качестве значения константу или структуру.
- **Константа** (constant) - это либо число (целое или действительное), либо атом, либо строка;
- **Операция** (operator) - это имя предиката или структуры, которое может быть записано в инфиксной, префиксной или постфиксной форме;
- **Переменная** (variable) - это одна из разновидностей термов Пролога. Значением переменной может быть константа или структура. Переменная обозначается словом, начинающимся с заглавной (английской) буквы;
- **Правило** (rule) - это фраза Хорна с одним или более условий;
- **Предикат** (predicate) - это отношение, существующее между некоторым фиксированным набором аргументов. Предикат идентифицируется по своему имени и количеству аргументов. В Прологе предикаты задаются при помощи процедур;
- **Процедура** (procedure) - это множество фраз, в котором все фразы имеют одно и тоже имя и одинаковое количество аргументов;
- **Подцель** (subgoal) - это условие в теле правила или в составном запросе;
- **Структура** (structure) - это сложный объект данных, идентифицируемый своим именем и количеством аргументов. Структура записывается как имя, за которым следует один или большее число аргументов, заключенных в скобки. Атом - это структура не имеющая аргументов;
- **Терм** (term) - это либо константа, либо переменная, либо структура;
- **Унификация** (unification) - это процесс согласования подцели с заголовком фразы при выполнении запроса. Подцель унифицируется с заголовком фразы, если выполняется три условия:
    - ♦ у них одинаковые имена;
    - ♦ они имеют одинаковое количество аргументов;
    - ♦ все их аргументы можно унифицировать. Унификация аргументов осуществляется в соответствии со следующими правилами:
        - ◊ две константы унифицируются друг с другом, если они идентичны;
        - ◊ переменная унифицируется с чем угодно (если с константой или структурой, то она конкретизируется, если с другой переменной, то они становятся одной и той же переменной);
    - ♦ структура унифицируется с другой структурой, если у них одинаковые имена, они имеют одинаковое количество аргументов и все их аргументы можно унифицировать;
- **Факт** - это фраза не имеющая условий;
- **Фраза** (clause) - это либо факт, либо правило, либо запрос.


Термы Пролога
Терм
Переменная
Константа
Структура
Анонимная
Число
Строка
Атом
Список
Веще-
ствен-
ное
Це-
лое