МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«КУБАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
Факультет компьютерных технологий и прикладной математики

УТВЕРЖДАЮ
Проректор по учебной работе, качеству образования – первый проректор
_____ Хагуров Т.А.
подпись
«29» августа 2025 г.

РАБОЧАЯ ПРОГРАММА ДИСЦИПЛИНЫ

ФТД.01 «Функциональное и логическое программирование»

Направление подготовки 01.03.02 Прикладная математика и информатика

Направленность (профиль) Современные методы машинного обучения и компьютерного зрения

Форма обучения очная

Квалификация бакалавр

Краснодар 2025

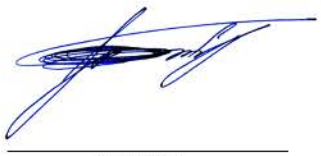
Рабочая программа дисциплины «Функциональное и логическое программирование» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования (ФГОС ВО) по направлению подготовки 01.03.02 Прикладная информатика.

Программу составил(и):
Подколзин В.В. канд. физ.-мат. наук, доцент

Рабочая программа дисциплины утверждена на заседании кафедры информационных технологий протокол № 1 от «26» августа 2025г.

Заведующий кафедрой Подколзин В.В.

Утверждена на заседании учебно-методической комиссии факультета компьютерных технологий и прикладной математики
протокол № 01 от «28» августа 2025 г.
Председатель УМК факультета
А. В. Коваленко



подпись

Рецензенты:

Мостовой Евгений Викторович, генеральный директор ООО «Портал-Юг», e-mail: mostovoy@portal-yug.ru

Луценко Евгений Вениаминович, доктор экономических наук, кандидат технических наук, профессор кафедры компьютерных технологий и систем Федерального государственного бюджетное образовательное учреждение высшего образования «Кубанский государственный аграрный университет имени И.Т. Трубилина», e-mail: prof.lutsenko@gmail.com

1 Цели и задачи изучения дисциплины (модуля)

1.1 Цель освоения дисциплины

Учебная дисциплина «Функциональное и логическое программирование» предназначена для изучения декларативной парадигмы программирования и её места в современной коммерческой и научной разработке.

Целью преподавания и изучения дисциплины «Функциональное и логическое программирование» является знакомство студентов с понятием парадигма программирования, изучение принципов работы в декларативном стиле, определение круга задач, решаемых модулями, написанными в императивной или декларативной парадигме, получение практических навыков писать читаемый код в функциональном или логическом стиле на актуальных языках программирования с применением современных платформ и фреймворков.

1.2 Задачи дисциплины

В результате освоения данной компетенции студент должен:

знать фундаментальные концепции написания программ в декларативном стиле, математические принципы лямбда исчисления, принципы функционального программирования, принципы логических переборных языков программирования.

уметь реализовывать модули анализа данных на основе функциональных интерфейсов, строить чистые функции высших порядков, реализовывать системы формального вывода и переборные алгоритмы средствами логического программирования, внедрять их в комплексные программные решения.

владеть навыками определения парадигмы, подходящей для решения конкретной задачи, навыками написания модулей работы с внешними системами (размеченные файлы, базы данных, потоки ввода) средствами языков функциональной и логической парадигмы программирования

1.3 Место дисциплины (модуля) в структуре образовательной программы

Дисциплина «Функциональное и логическое программирование» относится к дисциплинам факультативам.

Дисциплина в значительной степени **взаимодействует для формирования компетенций** с дисциплинами:

- Объектно-ориентированное программирование и шаблоны проектирования;
- Кроссплатформные десктоп приложения;
- Системы искуственного интеллекта

1.4 Профессиональные роли в структуре образовательной программы

Роль 1: **Data Engineer (Инженер по данным)**
Задачи:

- Проектирование и построение ETL-процессов
- Создание и оптимизация хранилищ данных
- Обеспечение качества и доступности данных
- Настройка инфраструктуры для обработки больших данных
- Интеграция разрозненных источников данных
- Работа с данными в области природопользования, медицины, связи и телекоммуникаций

Роль 2: **ML Engineer (Инженер МО)**
Задачи:

- Реализация ML-моделей в продуктивных системах
- Оптимизация производительности и масштабирование моделей
- Разработка ML-пайплайнов и автоматизация процессов
- Мониторинг качества моделей в продуктиве
- Интеграция ML-решений с бизнес-приложениями

Роль 3: **MLOps (Специалист по эксплуатации ИИ)**
Задачи:

- Автоматизация процессов обучения и развертывания моделей
- Мониторинг производительности ML-систем
- Управление версиями моделей и данных
- Обеспечение CI/CD для ML-проектов
- Оптимизация вычислительных ресурсов

1.5 Перечень планируемых результатов обучения по дисциплине (модулю), соотнесенных с планируемыми результатами освоения образовательной программы

Изучение данной учебной дисциплины направлено на формирование у обучающихся следующих компетенций: ПК-4.1; ПК-4.2; ПК-4.3; MF-6.1

ПК-4	***Способен планировать необходимые ресурсы и этапы выполнения работ в области информационно-коммуникационных технологий, составлять соответствующие технические описания и инструкции***
ПК-4.1	Использует современные инструментальные средства разработки баз данных, прикладного программного обеспечения и систем различного функционального назначения
	Знать классификацию и назначение современных инструментов разработки (IDE, системы контроля версий, СУБД, CI/CD) и критерии их выбора в зависимости от задач и технологического стека. Уметь выбирать, применять и настраивать инструменты на всех этапах жизненного цикла ПО — от проектирования до развертывания, а также интегрировать их в единый рабочий конвейер (pipeline). Владеть практическими навыками работы в ключевых инструментах выбранного стека и технологиями быстрого освоения новых средств, соблюдая best practices и культуру эффективного использования.
ПК-4.2	Применяет современные приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов на базе языков программирования, баз данных и пакетов прикладных программ
	Знать современные методологии, приёмы и best practices применения инструментальных средств (отладчики, профайлеры, системы сборки, фреймворки тестирования) для поддержки полного жизненного цикла ПО — от написания кода до отладки, оптимизации и сопровождения. Уметь осознанно применять специализированные приёмы работы с инструментами (например, интерактивная отладка, рефакторинг в IDE, написание автоматизированных тестов, конфигурирование сборок) для повышения качества, надёжности и эффективности процесса разработки программных продуктов и комплексов. Владеть комплексными навыками использования инструментальной экосистемы как единого процесса: от владения продвинутыми возможностями IDE и системами управления зависимостями до автоматизации развёртывания

	и мониторинга, обеспечивая воспроизводимость и промышленное качество результатов.
ПК-4.3	Способен использовать методы эффективного управления командой при разработке, внедрении и сопровождении программных продуктов
	Знать основные теории, модели и методы управления командой (Agile, Scrum, Kanban, фазы групповой динамики, ролевые модели) в контексте жизненного цикла ПО, а также принципы эффективной коммуникации, мотивации и разрешения конфликтов. Уметь применять эти методы на практике: планировать и распределять задачи (например, через бэклог спринта), проводить эффективные митинги (daily, ретроспективы), отслеживать прогресс командной работы, разрешать возникающие конфликты и адаптировать стиль управления под потребности проекта и этап его развития. Владеть навыками фасилитации командных процессов, использования современных инструментов управления проектами (Jira, Asana, Notion) и методами формирования психологически безопасной и продуктивной рабочей среды, направленной на достижение общих целей при разработке и сопровождении ПО.
MF-6	***Способен применять логический аппарат для формализации задач представления знаний, проектирования логических моделей и использования систем автоматического доказательства теорем.***
MF-6.1	Применяет логические структуры для принятия решений в автоматизированных системах ИИ Оптимизирует методы принятия решений с использованием формальных логических моделей.

2. Структура и содержание дисциплины

2.1 Распределение трудоёмкости дисциплины по видам работ

Общая трудоёмкость дисциплины составляет 2 зач. ед. (72 часа), их распределение по видам работ представлено в таблице

Вид учебной работы		**Всего часов**	**Семестры (часы)**					
			5					
Контактная работа, в том числе:		**52,2**	**52,2**					
Аудиторные занятия (всего):		**50**	**50**					
Занятия лекционного типа		**16**	16					
Лабораторные занятия		**34**	34					
Занятия семинарского типа (семинары, практические занятия)								
Иная контактная работа:		**2,2**	**2,2**					
Контроль самостоятельной работы (КСР)		**2**	2					
Промежуточная аттестация (ИКР)		**0,2**	0,2					
Самостоятельная работа, в том числе:		**19,8**	**19,8**					
Курсовая работа								
Выполнение индивидуальных заданий		**16**	16					
Реферат								
Подготовка к текущему контролю		**3,8**	**3.8**					
Контроль:								
Общая трудоемкость	**час.**	**72**	**72**					

	в том числе контактная работа	**52,2**	**52,2**					
	зач. ед	**2**	**2**					

2.2 Структура дисциплины

Распределение видов учебной работы и их трудоемкости по разделам дисциплины.

Разделы (темы) дисциплины, изучаемые в 5 семестре

№	Наименование разделов (тем)	Количество часов				
		Всего	Аудиторная работа			Внеаудиторная работа
			Л	ПЗ	ЛР	СРС
1	2	3	4	5	6	7
1.	**Раздел 1.** Основы логического программирования	22	6		10	6
2.	**Раздел 2.** Лямбда-исчисление.	16	4		8	4
3.	**Раздел 3.** Основы функционального программирования.	28	6		16	6
ИТОГО по разделам дисциплины		**66**	**16**		**34**	**16**
Контроль самостоятельной работы (КСР)		2				
Промежуточная аттестация (ИКР)		0,2				
Подготовка к текущему контролю		3,8				
Общая трудоемкость по дисциплине		**72**				

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

2.3 Содержание разделов (тем) дисциплины

2.3.1 Занятия лекционного типа

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
1.	**Раздел 1.** Основы логического программирования	Парадигмы программирования. Императивное программирование. Декларативное программирование. Особенности логического программирования и языка Prolog. Высказывания Общие принципы работы с SWI-Prolog Предикаты и правила Числовые задачи. Рекурсия. Построение дерева вывода.Унификация, бектрекинг и отсечение. Рекурсия вверх и вниз. Хвостовая рекурсия . Списки Черча. Общий подход. Реализация стандартных задач на списках Встроенные предикаты SWI-Prolog работы со списками Форматы представления строк. Работа с файлами. Динамические и статические факты и правила. Встроенные предикаты построения новых фактов или высказываний. Решение прикладных задач комбинаторики и теории графов средствами SWI-Prolog.	ЛР
2.	**Раздел 2.** Лямбда-исчисление.	Принципы и достоинства функционального программирования.	ЛР

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
		Чистая функция, функция высших порядков, неизменяемость данных, карринг, замыкание, неименованные функции. Принципы лямбда нотации. Понятие лямбда терма Свободные и связанные переменные. Подстановка и преобразования Лямбда исчисление. Эквивалентность лямбда выражений. Экстенсиональность Редукция Теорема Чёрча-Россера Комбинаторы. Лямбда исчисление как язык программирования. Представление данных в лямбда-исчислении Рекурсивные функции Let-выражения Достижение уровня полноценного языка программирования Списки Чёрча Типизированное лямбда исчисление Полиморфизм	
3.	**Раздел 3.** Основы функционального программирования.	Знакомство с Kotlin. Язык. Философия и инструментарий. Основные элементы: переменные и функции. Классы и свойства. Представление и обработка выбора: перечисления и конструкция «when» Итерации: циклы «while» и «for» Исключения в Kotlin Создание коллекций в Kotlin Определение и вызов функций Упрощение вызова функций Добавление методов в сторонние классы: функции-расширения и свойства-расширения Работа с коллекциями: переменное число аргументов, инфиксная форма записи вызова и поддержка в библиотеке Работа со строками и регулярными выражениями Классы, объекты и интерфейсы локальные функции и расширения Создание иерархий классов Объявление классов с нетривиальными конструкторами или свойствами Методы, сгенерированные компилятором: классы данных и делегирование Лямбда-выражения Совместное объявление класса и экземпляра Лямбда-выражения Функциональный API для работы с коллекциями Последовательности Система типов Лямбда-выражения с получателями Значение null Базовые	ЛР

№	Наименование раздела (темы)	Содержание раздела (темы)	Форма текущего контроля
1	2	3	4
		типы Массивы и коллекции Перегрузка операторов Перегрузка арифметических операторов Перегрузка операторов сравнения Коллекции и диапазоны Мультидекларации и функции Component Функции высшего порядка Делегирование свойств Объявление функций высших порядков Встраиваемые функции Порядок выполнения функций высших порядков Обобщенные типы Параметры обобщенных типов Стирание и овеществление параметров типов Обобщенные типы и подтипы Объявление и применение аннотаций Рефлексия: интроспекция объектов Kotlin во время выполнения Понятие предметно-ориентированного языка Внутренние предметно-ориентированные языки Структура предметно-ориентированных языков Создание разметки HTML с помощью внутреннего DSL Создание структурированных API: лямбда-выражения с получателями в DSL Гибкое вложение блоков с использованием соглашения «invoke» Предметно-ориентированные языки Kotlin на практик Сборка кода на Kotlin с помощью Gradle Сборка проектов на Kotlin с помощью Maven Сериализация JSON Клиенты HTTP Доступ к базам данных	

Примечание: ЛР – отчет/защита лабораторной работы, КП - выполнение курсового проекта, КР - курсовой работы, РГЗ - расчетно-графического задания, Р - написание реферата, Э - эссе, К - коллоквиум, Т – тестирование, РЗ – решение задач.

2.3.2 Занятия семинарского типа

Не предусмотрены

2.3.3 Лабораторные занятия

№ работы	№ раздела дисциплины	Наименование лабораторных работ	Форма текущего контроля
1	1	Введение в пролог, знакомство с логическим подходом на основе создания древовидных структур данных	ЛР1
2	1	Разработка рекурсивных предикатов для реализации числовых алгоритмов, работы со списками Черча, логических задач	ЛР2

3	1	Применение пролога для решения переборных задач комбинаторики и теории графов	ЛР3
4	1	Защита индивидуальных заданий	-
5	1	Контрольная работа № 1. Написание предикатов	КР1
6	2	Чистое лямбда исчисление.	-
7	2	Комбинаторы.	ЛР4
8	2	Реализация арифметических и логических функций в чистом λ-исчислении	-
9	2	Контрольная работа № 2. теоретические основы лямбда исчисления и систем формального вывода	КР2
10	3	Написание функций высших порядков в kotlin	ЛР5
11	3	Коллекции и высшие функции работы с ними	ЛР6
12	3	Анализ десериализованных данных с помощью механизмов функционального ООП	ЛР7
13	3	Защита индивидуальных заданий	-
14	3	Защита индивидуальных заданий	-
15	3	Защита индивидуальных заданий	-
16	3	Контрольная работа № 3. Написание функций высших порядков	КР3

2.3.4 Примерная тематика курсовых работ (проектов)

Курсовая работа не предусмотрена.

2.4 Перечень учебно-методического обеспечения для самостоятельной работы обучающихся по дисциплине (модулю)

№	Вид СРС	Перечень учебно-методического обеспечения дисциплины по выполнению самостоятельной работы
1	2	3
1	Изучение теоретического материала	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой вычислительных технологий, протокол №7 от 07.05.2025
2	Решение задач	Методические указания по организации самостоятельной работы студентов, утвержденные кафедрой вычислительных технологий, протокол №7 от 07.05.2025

Учебно-методические материалы для самостоятельной работы обучающихся из числа инвалидов и лиц с ограниченными возможностями здоровья (ОВЗ) предоставляются в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа,

– в форме аудиофайла,

– в печатной форме на языке Брайля.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа,

– в форме аудиофайла.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

3. Образовательные технологии

В соответствии с требованиями ФГОС в программа дисциплины предусматривает использование в учебном процессе следующих образовательные технологии: чтение лекций с использованием мультимедийных технологий; метод малых групп, разбор практических задач и кейсов.

При обучении используются следующие образовательные технологии:

– Технология коммуникативного обучения – направлена на формирование коммуникативной компетентности студентов, которая является базовой, необходимой для адаптации к современным условиям межкультурной коммуникации.

– Технология разноуровневого (дифференцированного) обучения – предполагает осуществление познавательной деятельности студентов с учётом их индивидуальных способностей, возможностей и интересов, поощряя их реализовывать свой творческий потенциал. Создание и использование диагностических тестов является неотъемлемой частью данной технологии.

– Технология модульного обучения – предусматривает деление содержания дисциплины на достаточно автономные разделы (модули), интегрированные в общий курс.

– Информационно-коммуникационные технологии (ИКТ) - расширяют рамки образовательного процесса, повышая его практическую направленность, способствуют интенсификации самостоятельной работы учащихся и повышению познавательной активности. В рамках ИКТ выделяются 2 вида технологий:

– Технология использования компьютерных программ – позволяет эффективно дополнить процесс обучения языку на всех уровнях.

– Интернет-технологии – предоставляют широкие возможности для поиска информации, разработки научных проектов, ведения научных исследований.

– Технология индивидуализации обучения – помогает реализовывать личностно-ориентированный подход, учитывая индивидуальные особенности и потребности учащихся.

– Проектная технология – ориентирована на моделирование социального взаимодействия учащихся с целью решения задачи, которая определяется в рамках профессиональной подготовки, выделяя ту или иную предметную область.

– Технология обучения в сотрудничестве – реализует идею взаимного обучения, осуществляя как индивидуальную, так и коллективную ответственность за решение учебных задач.

– Игровая технология – позволяет развивать навыки рассмотрения ряда возможных способов решения проблем, активизируя мышление студентов и раскрывая личностный потенциал каждого учащегося.

– Технология развития критического мышления – способствует формированию разносторонней личности, способной критически относиться к информации, умению отбирать информацию для решения поставленной задачи.

Комплексное использование в учебном процессе всех вышеназванных технологий стимулируют личностную, интеллектуальную активность, развивают познавательные процессы, способствуют формированию компетенций, которыми должен обладать будущий специалист.

Основные виды интерактивных образовательных технологий включают в себя:

– работа в малых группах (команде) - совместная деятельность студентов в группе под руководством лидера, направленная на решение общей задачи путём творческого сложения результатов индивидуальной работы членов команды с делением полномочий и ответственности;

– проектная технология - индивидуальная или коллективная деятельность по отбору, распределению и систематизации материала по определенной теме, в результате которой составляется проект;

– анализ конкретных ситуаций - анализ реальных проблемных ситуаций, имевших место в соответствующей области профессиональной деятельности, и поиск вариантов лучших решений;

– развитие критического мышления – образовательная деятельность, направленная на развитие у студентов разумного, рефлексивного мышления, способного выдвинуть новые идеи и увидеть новые возможности.

Подход разбора конкретных задач и ситуаций широко используется как преподавателем, так и студентами во время лекций, лабораторных занятий и анализа результатов самостоятельной работы. Это обусловлено тем, что при исследовании и решении каждой конкретной задачи имеется, как правило, несколько методов, а это требует разбора и оценки целой совокупности конкретных ситуаций.

Примечание: Л – лекции, ПЗ – практические занятия/семинары, ЛР – лабораторные занятия, СРС – самостоятельная работа студента

Темы, задания и вопросы для самостоятельной работы призваны сформировать навыки поиска информации, умения самостоятельно расширять и углублять знания, полученные в ходе лекционных и практических занятий.

Подход разбора конкретных ситуаций широко используется как преподавателем, так и студентами при проведении анализа результатов самостоятельной работы.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Для лиц с ограниченными возможностями здоровья предусмотрена организация консультаций с использованием электронной почты.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4. Оценочные и методические материалы

4.1 Оценочные средства для текущего контроля успеваемости и промежуточной аттестации

Оценочные средства предназначены для контроля и оценки образовательных достижений обучающихся, освоивших программу учебной дисциплины «название дисциплины».

Оценочные средства включает контрольные материалы для проведения **текущего контроля** в форме оценки лабораторных работ, контрольных работ, ответа на вопросы на зачёте и проекта к **зачету**.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине (модулю) предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

Структура оценочных средств для текущей и промежуточной аттестации

№ п/п	Контролируемые разделы (темы) дисциплины*	Код контролируемой компетенции (или ее части)	Наименование оценочного средства	
			Текущий контроль	Промежуточная аттестация
1	**Раздел 1.** Основы логического программирования	ПК-4.1; ПК-4.2; ПК-4.3; MF-6.1	*ЛР 1-3, КР-1*	*Зачетный проект, вопросы по разделу*
2	**Раздел 2.** Лямбда-исчисление.	ПК-4.1; ПК-4.2; ПК-4.3	*ЛР 4, КР-2*	*Зачетный проект, вопросы по разделу*
3	**Раздел 3.** Основы функционального программирования.	ПК-4.1; ПК-4.2; ПК-4.3	*ЛР 5-7, КР-3*	*Зачетный проект, вопросы по разделу*

Показатели, критерии и шкала оценки сформированных компетенций

Соответствие **базовому уровню** освоения компетенций планируемым результатам обучения и критериям их оценивания (оценка: **зачтено**):

ПК-4	***Способен планировать необходимые ресурсы и этапы выполнения работ в области информационно-коммуникационных технологий, составлять соответствующие технические описания и инструкции***

ПК-4.1	Использует современные инструментальные средства разработки баз данных, прикладного программного обеспечения и систем различного функционального назначения ***знать*** фундаментальные концепции написания программ в декларативном стиле, математические принципы лямбда исчисления, принципы функционального программирования, принципы логических переборных языков программирования
ПК-4.2	Применяет современные приемы работы с инструментальными средствами, поддерживающими создание программных продуктов и программных комплексов на базе языков программирования, баз данных и пакетов прикладных программ **уметь** реализовывать модули анализа данных на основе функциональных интерфейсов, строить чистые функции высших порядков, реализовывать системы формального вывода и переборные алгоритмы средствами логического программирования, внедрять их в комплексные программные решения
ПК-4.3	Способен использовать методы эффективного управления командой при разработке, внедрении и сопровождении программных продуктов **владеть** навыками определения парадигмы, подходящей для решения конкретной задачи, навыками написания модулей работы с внешними системами (размеченные файлы, базы данных, потоки ввода) средствами языков функциональной и логической парадигмы программирования
MF-6	***Способен применять логический аппарат для формализации задач представления знаний, проектирования логических моделей и использования систем автоматического доказательства теорем.***
MF-6.1	Применяет логические структуры для принятия решений в автоматизированных системах ИИ Оптимизирует методы принятия решений с использованием формальных логических моделей.

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

Требования к ЗП:

Олимпиадные задания.

Необходимо решить 5 олимпиадных задач.

Для каждой задачи необходимо:

FR<№задачи>.1. составить эффективный алгоритм с выполнением требований по времени и памяти на языке java;

FR<№задачи>.2. составить подробное описание решения;

FR<№задачи>.3. подготовить 10 нетривиальных тестов с конкретными исходными данными;

FR<№задачи>.4. провести тестирование;

FR<№задачи>.5. записать результаты тестирования, указав время работы и использованную память, ответы должны совпасть с тест кейсами;

FR<№задачи>.6. составить решение в логическом стиле на языке prolog;

FR<№задачи>.7. составить подробное описание решения;

FR<№задачи>.8. провести тестирование на подготовленных данных, ответы должны совпасть с тест кейсами;

FR<№задачи>.9. составить решение полностью в функциональном стиле без использования циклов и переменных на языке Kotlin;

FR<№задачи>.10. составить подробное описание решения;

FR<№задачи>.11. провести тестирование на подготовленных данных, ответы должны совпасть с тест кейсами.

Нефункциональные требования – решения должны быть на общем для команды репозитории, тесты там же, должна быть видна история коммитов, задача в один коммит не принимается.

Требования по оцениванию каждого из параметров:

- логическая часть,
- императивная часть,
- функциональная часть,
- интеграция в готовое решение.

Императивная часть

На оценку + необходимо выполнить требования:

FR1.1.
FR1.3.
FR1.4.
FR2.1.
FR2.3.
FR2.4.
FR2.5.

На оценку Удовлетворительно необходимо выполнить требования:

требования на +
FR3.1.
FR3.3.
FR3.4.

На оценку Хорошо необходимо выполнить требования:

требования на удовлетворительно
FR4.1.
FR4.3.
FR4.4.

На оценку Отлично необходимо выполнить требования:

требования на хорошо
FR5.1.
FR5.3.
FR5.4.

Логическая часть.

На оценку + необходимо выполнить требования:

требования на + императивной части

FR1.6.

FR1.8.

FR2.6.

FR2.8.

На оценку Удовлетворительно необходимо выполнить требования:

требования на + логической части

требования на Удовлетворительно императивной части

FR3.6.

FR3.8.

На оценку Хорошо необходимо выполнить требования:

требования на удовлетворительно логической части

требования на хорошо императивной части

FR4.6.

FR4.8.

На оценку Отлично необходимо выполнить требования:

требования на хорошо логической части

требования на отлично императивной части

FR5.6.

FR5.8.

Функциональная часть.

На оценку + необходимо выполнить требования:

требования на + императивной части

FR1.9.

FR1.11.

FR2.9.

FR2.11.

На оценку Удовлетворительно необходимо выполнить требования:

требования на + функциональной части

требования на Удовлетворительно императивной части

FR3.9.

FR3.11.

На оценку Хорошо необходимо выполнить требования:

требования на удовлетворительно функциональной части

требования на хорошо императивной части

FR4.9.

FR4.11.

На оценку Отлично необходимо выполнить требования:

требования на хорошо функциональной части

требования на отлично императивной части

FR5.9.

FR5.11.

Интеграционная часть.
Для олимпиадных задач интеграционная часть заключается в подготовке подробных описаний решений с объяснениями и структурой программы
После того, как задача корректно решена в любом стиле, можно оформлять документацию.
Документация должна быть согласована с комиссией преподавателей дисциплины. Документация оформляется в week, отметка о согласовании получается там же.
Всего необходимо выполнить 20 требований:
FR1.2.
FR1.5.
FR1.7.
FR1.10.
FR2.2.
FR2.5.
FR2.7.
FR2.10.
FR3.2.
FR3.5.
FR3.7.
FR3.10.
FR4.2.
FR4.5.
FR4.7.
FR4.10.
FR5.2.
FR5.5.
FR5.7.
FR5.10.
Оценка + выставляется в случае выполненных 8 требований
Оценка 3 выставляется в случае выполненных 12 требований
Оценка 4 выставляется в случае выполненных 16 требований
Оценка 5 выставляется в случае выполненных 20 требований

Требования к докладу.
Время на доклад 15 минут.
Во время доклада отобразить – гит с историей выполнения, согласованную документацию, постановку решенных задач, несколько нетривиальных тест кейсов, краткое пояснение задумки в решении.
Время на вопросы к докладу – 5 минут.

***Акинатор*:**

Требования к акинатору.

Должна быть выбрана конкретная область для объектов, количество объектов не менее 150 штук, количество вопросов – не менее 8.

FR1.1. Подготовлено приложение Prolog Акинатор, вызываемый из терминала Prolog, позволяющий угадывать персонажа и добавлять персонажа, если не существует объекта, имеющего полученную последовательность ответов на вопросы. Реализована возможность выдачи ответа пользователю в случае неполного ответа на вопросы, например, если лишь объект X имеет текущую картину ответа на 5 вопросов, значит 6 и 7 вопрос задавать необязательно. Реализована возможность разных вопросов для разных объектов, например, 6 вопросов одинаковых, но, чтобы отличить объект 18 от объекта19 нужен вопрос 7, а чтобы отличить объект20 от объекта21, нужен вопрос 8, а остальные объекты отличаются друг от друга на основании первых 6 вопросов. Структура должна быть подобрана исходя из предметной области.

FR1.2. Продумать и построить http сервис,

- получающий ответ на текущий вопрос и возвращающий следующий вопрос или объект или результат об отсутствии объекта;
- добавляющий новый объект в случае отсутствия такового.

FR1.3. Модифицировать сервис, реализовав следующую возможность:

пусть есть два объекта X1 и X2, имеющих одинаковые ответы на все заданные вопросы, и пусть X1 уже в базе а X2 – нет. Если пользователь загадал X2, дать ему возможность добавить X2 в базу, дать ему возможность добавить вопрос, по которому можно отличить X1 от X2 с вариантами ответа.

FR1.4. Придумать и реализовать дополнительную фичу в rest сервисе.

FR2.1. Построить БД, содержащую информацию о пользователях и их сеансах игры в акинатор, в частности – загаданные персонажи, ответы на вопросы, добавленные персонажи.

FR2.2. Построить Rest сервис java spring, позволяющий добавлять пользователей, информацию о сеансах игры, включая – дату время игры, загаданных персонажей, ответы на вопросы, добавленных персонажей.

FR2.3. Модифицировать сервис и БД, добавив возможность добавлять вопросы и ответы на вопросы, сохраняя информацию о пользователе, добавившем вопрос и ответ, согласовав данную модификацию с требованиями FR1.3.

FR2.4. Поддержать сохранение информации о функционале, реализованном в FR1.4.

FR3.1. Построить два содержательных имеющих смысл запроса к БД из FR2.1 на языке kotlin с использованием функционального интерфейса о работе пользователей с акинатором.

FR3.2. Построить rest сервис с использованием kotlin spring, позволяющий выполнять построенные ранее два запроса.

FR3.3. Построить два дополнительных нетривиальных запроса с использованием информации из FR2.3.

FR3.4. Построить дополнительный запрос с использованием информации из FR2.4.

FR4.1. Построить Java swing desktop приложение, реализующее игру в акинатор, с использованием функционала, выполненного в FR1.1, FR2.1, FR3.1. Обязательна авторизация пользователя и сохранение всей информации о сеансе игры. Важно, любая собираемая и отображаемая статистика должна быть результатами ваших запросов из FR3.1, запросы – отдельно, отображаемая статистика – отдельно – такая ситуация запрещена.

FR4.2. Построить web приложение, реализующее игру в акинатор с использованием http сервисов из FR1.2, FR2.2, FR3.2. Обязательна авторизация пользователя и сохранение всей информации о сеансе игры. Важно, любая собираемая и отображаемая статистика должна быть результатами ваших запросов из FR3.1, запросы – отдельно, отображаемая статистика – отдельно – такая ситуация запрещена.

FR4.3. Реализовать возможность добавления вопросов в соответствии с FR1.3., FR2.3., FR3.3.

FR4.4. Реализовать фичу после FR1.4, FR2.4, FR3.4

FR4.5. Отобразить полное дерево предметной области с ответами на вопросы и объектами (или сохраняя изображение с крупным разрешением и возможностью масштабирования или отображая рисунок в приложении с возможностью масштабирования и пролистывания в масштабе).

Требования к ведению проекта.

Все сервисы должны быть подняты на ресурсах лаборатории КубГУ.

Все изменения сохраняются только на гитлабе лаборатории КубГУ. Реализация любого требования – больше одного коммита. У любого участника команды – больше одного коммита. На http сервисы прописаны автотесты и проведены автотесты, сохранены отчёты. На каждый сервис подготовлен swagger и доступен на ресурсах лаборатории КубГУ.

Логическая часть

На оценку + необходимо выполнить требования:

FR1.1.

На оценку Удовлетворительно необходимо выполнить требования:

FR1.2.

На оценку Хорошо необходимо выполнить требования:

требования на удовлетворительно логической части

или FR1.3. или FR1.4.

На оценку Отлично необходимо выполнить требования:

требования на удовлетворительно логической части

FR1.3.

FR1.4.

Императивная часть

На оценку + необходимо выполнить требования:

требования на + логической части
FR2.1.

На оценку Удовлетворительно необходимо выполнить требования:
требования на удовлетворительно логической части
FR2.2.

На оценку Хорошо необходимо выполнить требования:
требования на хорошо логической части
требования на удовлетворительно императивной части
или FR2.3. или FR2.4.

На оценку Отлично необходимо выполнить требования:
требования на отлично логической части
требования на удовлетворительно императивной части
FR2.3.
FR3.4.

Функциональная часть

На оценку + необходимо выполнить требования:
требования на + императивной части
FR3.1.

На оценку Удовлетворительно необходимо выполнить требования:
требования на удовлетворительно императивной части
FR3.2.

На оценку Хорошо необходимо выполнить требования:
требования на хорошо императивной части
требования на удовлетворительно функциональной части
или FR3.3. или FR3.4.

На оценку Отлично необходимо выполнить требования:
требования на отлично императивной части
требования на удовлетворительно функциональной части
FR3.3.
FR3.4.

Интеграционная часть

На оценку + необходимо выполнить
требования на + логической, императивной и функциональной части
FR4.1

На оценку Удовлетворительно необходимо выполнить
требования на удовлетворительно логической, императивной и функциональной части
FR4.2

Или

На оценку Удовлетворительно необходимо выполнить

требования на + интеграционной части
FR4.5

На оценку Хорошо необходимо выполнить
требования на хорошо логической, императивной и функциональной части
FR4.3 или FR4.4.

или

На оценку Хорошо необходимо выполнить
требования на удовлетворительно интеграционной части
FR4.5

На оценку Отлично необходимо выполнить
требования на отлично логической, императивной и функциональной части
FR4.3
FR4.4

Или
На оценку Отлично необходимо выполнить
требования на хорошо интеграционной части
FR4.5

Требования к докладу

Время на доклад 15 минут.
Во время доклада отобразить – гит с историей выполнения, продемонстрировать работу ПО, продемонстрировать реализацию каждого из требований.
Прислать перед выступлением ссылки на все сервисы, web приложения и swagger каждого сервиса.
Время на вопросы к докладу – 5 минут.

***Логическая игра*:**

Нельзя – крестики нолики 3 на 3.

FR1.1. Подготовлено приложение Prolog, вызываемое из терминала Prolog, позволяющее играть в игру с компьютером.

FR1.2. Модифицировать приложение, реализовав возможность установить 2 уровня сложности.

FR1.3. Продумать и построить http сервис:

- выполняющий действие – сделай ход;
- проверяющий ситуацию на победу игрока или бота.

FR1.4. Модифицировать сервис, реализовав возможность установить 2 уровня сложности.

FR1.5. Модифицировать сервис, реализовав возможность установить 4 уровня сложности.

FR1.6. Модифицируйте сервис, придумав свою собственную фичу в ведении игры ботом.

FR1.7. Модифицируйте сервис, придумав способ определения того, кто ближе к победе по текущему состоянию игры.

FR2.1. Построить БД, содержащую информацию о пользователях и их сеансах игры, в частности – победы, поражения,

FR2.2. Модифицировать БД, сохранив информацию об уровнях сложности игры, обязательно после FR1.2.

FR2.3. Построить Rest сервис java spring, позволяющий добавлять пользователей, информацию о сеансах игры, включая – дату время игры, победы, поражения, текущий сеанс игры, получать текущий сеанс игры, обязательно после FR1.3.

FR2.4. Модифицировать сервис и БД, добавив возможность сохранять информацию об уровне сложности, обязательно после FR1.4 или FR1.5.

FR2.5. Модифицировать сервис и БД, добавив возможность сохранять и получать информацию об нескольких сеансах игры одним пользователем.

FR2.6. Модифицировать сервис и БД, реализовав фичу из 2.6.

FR2.7. Модифицировать сервис и БД, добавив возможность предварительного завершения игры и сохранения информации об этом, обязательно после FR1.7.

FR3.1. Построить два содержательных имеющих смысл запроса к БД из FR2.1 на языке kotlin с использованием функционального интерфейса о работе пользователей с игрой, только после реализованной БД из 2.1.

FR3.2. Построить rest сервис с использованием kotlin spring, позволяющий выполнять построенных ранее два запроса.

FR3.3. Построить два дополнительных нетривиальных запроса из реализованного функционала 2.4 – 2.7.

FR3.4. Построить дополнительный сложный запрос из реализованного функционала 2.4 – 2.7.

FR4.1. Построить Java swing desktop приложение, реализующее игру, с использованием функционала, выполненного в FR1.1, FR2.1, FR3.1.

Обязательна авторизация пользователя и сохранение всей информации о сеансе игры. Важно, любая собираемая и отображаемая статистика должна быть результатами ваших запросов из FR3.1, запросы – отдельно, отображаемая статистика – отдельно – такая ситуация запрещена.

FR4.2. Построить Java swing desktop приложение, реализующее игру, с использованием функционала, выполненного в FR1.2, FR2.2, FR3.1. Обязательна авторизация пользователя и сохранение всей информации о сеансе игры. Важно, любая собираемая и отображаемая статистика должна быть результатами ваших запросов из FR3.1, запросы – отдельно, отображаемая статистика – отдельно – такая ситуация запрещена.

FR4.3. Построить web приложение, реализующее игру в с использованием http сервисов из FR1.3, FR2.3, FR3.2. Обязательна авторизация пользователя и сохранение всей информации о сеансе игры, сохранение и продолжение игры пользователем. Важно, любая собираемая и отображаемая статистика должна быть результатами ваших запросов из FR3.2, запросы – отдельно, отображаемая статистика – отдельно – такая ситуация запрещена.

FR4.4. Модифицировать web приложение, реализовав уровни сложности, обязательно вместе (FR1.4 или FR1.5) и (FR2.4).

FR4.5. Модифицировать web приложение, реализовав возможность ведения нескольких игр, обязательно вместе с FR2.5

FR4.6. Модифицировать web приложение, реализовав свою фичу, обязательно вместе с FR1.6 и FR2.6.

FR4.7. Модифицировать web приложение, реализовав возможность предварительного завершения игры, обязательно вместе с FR1.7 и FR2.7.

FR4.8. Отобразить результаты 3.3 в web приложении.

FR4.9. Отобразить результаты 3.4 в web приложении.

Требования к ведению проекта.

Все сервисы должны быть подняты на ресурсах лаборатории КубГУ.

Все изменения сохраняются только на гитлабе лаборатории КубГУ. Реализация любого требования – больше одного коммита. У любого участника команды – больше одного коммита. На http сервисы прописаны автотесты и проведены автотесты, сохранены отчёты. На каждый сервис подготовлен swagger и доступен на ресурсах лаборатории КубГУ.

Логическая часть

На оценку + необходимо выполнить требования:

FR1.1.

На оценку Удовлетворительно необходимо выполнить требования:

FR1.2. или FR1.3

На оценку Хорошо необходимо выполнить требования:

FR1.3

FR1.4 или FR1.6 или FR1.7

На оценку Отлично необходимо выполнить требования:

FR1.3

FR1.5

Или

На оценку Отлично необходимо выполнить требования:

FR1.3

FR1.4

FR1.6

или

FR1.3

FR1.4

FR1.7

или

FR1.3

FR1.6

FR1.7

Императивная часть

На оценку + необходимо выполнить требования:

FR1.1

FR2.1.

На оценку Удовлетворительно необходимо выполнить требования:

FR2.2.

или

На оценку Удовлетворительно необходимо выполнить требования:

FR2.3.

На оценку Хорошо необходимо выполнить требования:

FR2.3

FR2.4 или FR2.5 или FR2.6 или FR2.7

На оценку отлично необходимо выполнить

FR2.3

Любые два из требований FR2.4 или FR2.5 или FR2.6 или FR2.7

Функциональная часть

На оценку + необходимо выполнить требования:

FR3.1.

На оценку Удовлетворительно необходимо выполнить требования:

требования на удовлетворительно императивной части

FR3.2.

На оценку Хорошо необходимо выполнить требования:

требования на хорошо императивной части

требования на удовлетворительно функциональной части
FR3.3.

На оценку Отлично необходимо выполнить требования:
требования на отлично императивной части
требования на хорошо функциональной части
FR3.4.

Интеграционная часть

На оценку + необходимо выполнить
требования на + логической, императивной и функциональной части
FR4.1

На оценку Удовлетворительно необходимо выполнить
требования на удовлетворительно логической, императивной части,
требования на + функциональной части
FR4.2

Или

На оценку Удовлетворительно необходимо выполнить
FR4.3

На оценку Хорошо необходимо выполнить
FR4.3
FR4.8
FR4.4 или FR4.5 или FR4.6 или FR4.7

На оценку Отлично необходимо выполнить
FR4.3
FR4.8
FR4.9
Любые два из требований FR2.4 или FR2.5 или FR2.6 или FR2.7

Требования к докладу

Время на доклад 15 минут.
Во время доклада отобразить – гит с историей выполнения, продемонстрировать работу ПО, продемонстрировать реализацию каждого из требований.
Прислать перед выступлением ссылки на все сервисы, web приложения и swagger каждого сервиса.
Время на вопросы к докладу – 5 минут.

БД:

Требования к БД.

FR2.1. Подготовить БД из 3-4 таблиц в 3НФ на любую предметную область, заполнено миниму по 20 записей в КАЖДОЙ таблице.

FR1.1. Реализовать prolog БД, дублирующую БД из FR1.1.

FR2.2. Построить java spring REST сервис, реализующий CRUD для каждой из таблиц.

FR1.2. Построить http prolog сервис, реализующий CRUD для каждой из таблиц.

FR3.1. Построить сложный запрос с использованием всех таблиц с применением функционального интерфейса kotlin.

FR3.2. Построить kotlin spring REST сервис, реализующий этот запрос.

FR4.1. Построить java swing приложение реализующее CRUD на 2.1 и 1.1, с возможностью выбора, где работать, в БД или прологе, и отобразить результаты 3.1.

FR4.2. Построить web приложение реализующее CRUD на 2.2 и 1.2, с возможностью выбора, где работать, в БД или прологе, и отобразить результаты 3.2.

FR4.3. Модифицировать web приложение и сервисы так, чтобы была возможность синхронизации из prolog в БД.

FR4.4. Модифицировать web приложение и сервисы так, чтобы была возможность синхронизации из БД в prolog.

FR3.3. Модифицировать kotlin spring REST сервис: Построить 2 доп запроса с использованием всех таблиц с применением функционального интерфейса kotlin.

FR3.4. Модифицировать kotlin spring REST сервис: Построить 2 доп запроса с использованием всех таблиц с применением функционального интерфейса kotlin.

FR4.6. Отобразить результаты 3.3 в web приложении.

FR4.7. Отобразить результаты 3.4 в web приложении.

Требования к ведению проекта.

Все сервисы должны быть подняты на ресурсах лаборатории КубГУ.

Все изменения сохраняются только на гитлабе лаборатории КубГУ. Реализация любого требования – больше одного коммита. У любого участника команды – больше одного коммита. На http сервисы прописаны автотесты и проведены автотесты, сохранены отчёты. На каждый сервис подготовлен swagger и доступен на ресурсах лаборатории КубГУ.

Логическая часть

На оценку + необходимо выполнить требования:

FR1.1.

На оценку Удовлетворительно необходимо выполнить требования:

FR1.2.

На оценку Хорошо необходимо выполнить требования:
FR4.3 или FR4.4

На оценку Отлично необходимо выполнить требования:
FR4.3
FR4.4

Императивная часть

На оценку + необходимо выполнить требования:
FR2.1.
На оценку Удовлетворительно необходимо выполнить требования:
FR2.2.
На оценку Хорошо необходимо выполнить требования:
FR4.3 или FR4.4

На оценку Отлично необходимо выполнить требования:
FR4.3
FR4.4

Функциональная часть

На оценку + необходимо выполнить требования:
FR3.1.
На оценку Удовлетворительно необходимо выполнить требования:
FR3.2.
На оценку Хорошо необходимо выполнить требования:
FR3.3.
На оценку Отлично необходимо выполнить требования:
FR3.4.

Интеграционная часть

На оценку + необходимо выполнить
FR4.1

На оценку Удовлетворительно необходимо выполнить
FR4.2

На оценку Хорошо необходимо выполнить
FR4.2
Любые два из требований FR4.3 или FR4.4 или FR4.5 или FR4.6

На оценку Отлично необходимо выполнить
FR4.2
FR4.3
FR4.4.

FR4.5.
FR4.6

Требования к докладу

Время на доклад 15 минут.
Во время доклада отобразить – гит с историей выполнения, продемонстрировать работу ПО, продемонстрировать реализацию каждого из требований.
Прислать перед выступлением ссылки на все сервисы, web приложения и swagger каждого сервиса.

Время на вопросы к докладу – 5 минут

Контрольная работа № 1
по дисциплине
«Функциональное и логическое программирование»
направления 02.03.02 ФИИТ ФКТиПМ
на тему «Построение предикатов на языке SWI-Prolog»
Вариант № 1.

- Общая структура фактов в прологе. Составные факты.
- На примере числовых алгоритмов объясните смысл рекурсии вверх и рекурсии вниз в прологе.
- Построить предикат maxList(+List, Ind), определяющий индекс элемента списка, имеющего максимальную сумму цифр. (Воспользоваться рекурсией вверх или вниз).
- Построить предикат comb(+List,K,-Sochet), записывающий в Sochet все возможные сочетания по K элементов. (Возможна формулировка для любого комбинаторного объекта, размещения, перестановки, подмножества, с размещениями или без).
- Построить предикат, который выводит на экран все слова длины 6 над алфавитом [a,b,c,d,e,f], в которых три буквы a, две буквы b.
- Граф задан списком вершин и списком ребер, где каждое ребро – список двух вершин. [a,b,c,d,e], [[a,b],[a,e],[b,c],[c,d],[d,e],[c,e],[a,c]].

  Зная, что таким образом задан ОРИЕНТИРОВАННЫЙ граф, найти эйлеров цикл. (или любые задачи на графы из лекции)

Контрольная работа № 1
по дисциплине
«Функциональное и логическое программирование»
направления 02.03.02 ФИИТ ФКТиПМ
на тему «Построение предикатов на языке SWI-Prolog»
Вариант № 2.

1. Бектрекинг. Маркеры. Оператор отсечения.
2. Опишите принцип работы со списками Черча в Swi-prolog. Покажите реализации встроенных предикатов работы со списками на основе механизма унификации(append, reverse nth0).
3. Построить предикат fib(+N,?A), проверяющий является ли число A числом Фибоначчи с номером N или находящий число с таким номером и записывающим его в A. (любой числовой алгоритм с применением рекурсии вверх или вниз).
4. Построить предикат razm(+List,K,-Razm), записывающий в Razm все возможные

размещения по K элементов без повторений. Построить алгоритм на языке Java, выводящий на экран все размещения заданного списка по k элементов без повторений – (в случае, если задача 3 – простая, в задаче 4 включается требование реализации одной и той же задачи в императивном и логическом стиле).

5. Построить предикат, который выводит на экран все слова длины 6, в которых первые три буквы любые из [a,b,c,d,e] без повторов, остальные буквы [v,w,x,y,z] возможно с повторами.
6. Граф задан списком вершин и списком ребер, где каждое ребро – список двух вершин. [a,b,c,d,e], [[a,b],[a,e],[b,c],[c,d],[d,e],[c,e],[a,c]].

   Зная, что таким образом задан НЕОРИЕНТИРОВАННЫЙ граф, найти гамильтонов цикл. (или любые задачи на графы из лекции).

Контрольная работа № 2
по дисциплине
«Функциональное и логическое программирование»
направления 02.03.02 ФИИТ ФКТиПМ
на тему «Лямбда исчисление»
Вариант № 0.

1. Дайте определение редукции лямбда термов
2. Типизация по Чёрчу.
3. Редуцировать терм аппликативным и нормальным порядком

$$(\lambda xuz.\, x(\lambda x.\, zu))((\lambda x.\, u(xx))(\lambda x.\, u(xx)))$$

4. Проверить, что данный терм является комбинатором неподвижной точки

$$\mathbf{SI}(\mathbf{SII}(\mathbf{B}(\mathbf{SI})(\mathbf{SII})))$$

5. Составить комбинаторы, позволяющие реализовать следующие функции

   1. $x_1 \vee x_2 \vee x_3$;
   2. $x_1x_2 \vee x_1x_3 \vee x_2x_3$;
   3. $n - 2$;
   4. n^3;

6. Построить комбинатор F, обладающий следующим свойством для произвольных λ-термов X, Y, Z:

$$FXYZ = F(FX)(FXY)(FXYZ)$$

Контрольная работа № 2
по дисциплине
«Функциональное и логическое программирование»
направления 02.03.02 ФИИТ ФКТиПМ
на тему «Лямбда исчисление»
Вариант № 0.

7. Сформулируйте и докажите лемму о комбинаторах I, K, S
8. Let выражения
9. Дан терм, найти свободные и связанные переменные.

$$\big((\lambda\, u.\, (\lambda\, x.\, uxx)(\lambda\, x.\, uxy)\big)(\lambda\, y.\, y)$$

10. Редуцировать терм аппликативным и нормальным порядком

$$\mathbf{CI}(\mathbf{SB})\big(\mathbf{SB}(\mathbf{KI})\big)\big(\mathbf{SB}\big(\mathbf{SB}(\mathbf{KI})\big)\big)$$

11. Проверить, что данный терм является комбинатором неподвижной точки

$(\lambda xyz.\, y(xyy))\,(\lambda xyz.\, y(xyy))\,(\lambda xyz.\, y(xyy))$

12. Редуцировать термы: Add 5 2; Inc 3; Dec 3; Mult 4 2; Or (and true false) (and true true).

Контрольная работа № 3
по дисциплине
«Функциональное и логическое программирование»
направления 02.03.02 ФИИТ ФКТиПМ
на тему «Построение рекурсивных функций и функций высших порядков»
Вариант № 0.

13. Объясните смысл термина функции высших порядков. Приведите примеры.
14. Ассоциативные массивы. Создания и принципы работы.
15. Реализовать функцию, вычисляющую количество делителей числа n с помощью рекурсии вверх или вних
16. Построить функцию, которая принимает три аргумента
    - список
    - предикат p : int -> bool
    - функцию f : int -> int

    И возвращает список, состоящий из значений f от тех элементов, которые удовлетворяют предикату p.
17. Реализовать с помощью функции из задания 4 функцию, которая принимает список и возвращает новый список, состоящий из количеств делителей простых положительных элементов списка.
18. Для введенного списка построить новый список, который получен из начального упорядочиванием по количеству встречаемости элемента,

    То есть из списка [5,6,2,2,3,3,3,5,5,5] необходимо получить список [5,5,5,5,3,3,3,2,2,6].

Лямбда-исчисление

1. Опишите основные принципы декларативного программирования и его отличия от императивного. Опишите основные принципы функционального программирования и вытекающие из них преимущества и недостатки
2. Опишите понятия высшая функция, чистая функция, каррирование и ленивые вычисления. Опишите математические предположения, которые привели к лямбда исчислению и объясните формат записи лямбда выражений.
3. Дайте определение лямбда терма. Сформулируйте понятия абстракции и аппликации. Опишите соглашения о возможности опускать скобки, принятые в лямбда выражении. На примере лямбда термов опишите принцип каррирования и его работы.
4. Дайте определения свободных и связанных переменных в лямбда термах.
5. Дайте определения редукции лямбда термов. Опишите стратегии редукции лямбда термов.
6. Дайте понятия подстановки и преобразования. Сформулируйте понятия эквивалентности.
7. Сформулируйте теорему Черча-Россера, сформулируйте и докажите следствия из теоремы Черча-Россера.
8. Сформулируйте и докажите лемму о комбинаторах I, K, S, Докажите, что любой терм представим в виде комбинаторов S K

9. Понятие и виды комбинаторов. Редукция комбинаторов. Приведите примеры, покажите связь с лямбда исчислением.

10. Числа Черча. Операция плюс 1. Арифметические операции над числами Черча + * ^, -1.

11. Булевы константы и оператор if, Реализация булевых операций.

12. Кортежи. Каррирование. Их связь в лямбда-исчислении.

13. Комбинатор неподвижной точки. Рекурсивные функции (на примере любой функции).

14. Let выражения. Реализация списков Черча.

15. Полнота лямбда исчисления по Тьюрингу

16. Типизация по Черчу.

17. Типизация по Карри. Полиморфизм в типизации по Карри

18. Сформулируйте и докажите леммы и теорему о сохранении типов в типизации по Карри.

19. Сформулируйте недостатки математических моделей типизации лямбда по Карри и Черчу. Сформулируйте теорему о сильной нормализации

20. Покажите способ реализации типизации для генераторов неподвижной точки и рекурсии.

Типовые формулировки задач

№ 1. Дан терм, представленный в виде аппликации и абстракции лямбда-термов. Определить свободные и связанные переменные, провести редукцию с помощью двух возможных стратегий.

№ 2. Дан терм, представленный в виде аппликации комбинаторов, провести редукцию двумя способами.

№ 3. Дан терм, проверить, является ли он комбинатором неподвижной точки.

№ 4. Составить комбинаторы, позволяющие реализовать указанные функции (числовые или алгебры логики)

№ 5. Провести редукцию арифметических и логических выражений.

Функциональное программирование на примере языка kotlin.

1. Знакомство с Kotlin. Язык. Философия и инструментарий.
2. Основные элементы: переменные и функции. Классы и свойства.
3. Представление и обработка выбора: перечисления и конструкция «when»
4. Итерации: циклы «while» и «for»
5. Исключения в Kotlin
6. Создание коллекций в Kotlin
7. Определение и вызов функций
8. Упрощение вызова функций
9. Добавление методов в сторонние классы: функции-расширения и свойства-расширения
10. Работа с коллекциями: переменное число аргументов, инфиксная форма записи вызова и поддержка в библиотеке
11. Работа со строками и регулярными выражениями
12. Классы, объекты и интерфейсы локальные функции и расширения
13. Создание иерархий классов
14. Объявление классов с нетривиальными конструкторами или свойствами
15. Методы, сгенерированные компилятором: классы данных и делегирование
16. Лямбда-выражения
17. Совместное объявление класса и экземпляра
18. Лямбда-выражения Функциональный API для работы с коллекциями

19. Последовательности
20. Система типов
21. Лямбда-выражения с получателями
22. Значение null Базовые типы
23. Массивы и коллекции Перегрузка операторов Перегрузка арифметических операторов
24. Перегрузка операторов сравнения
25. Коллекции и диапазоны
26. Мультидекларации и функции
27. Component
28. Функции высшего порядка
29. Делегирование свойств
30. Объявление функций высших порядков
31. Встраиваемые функции
32. Порядок выполнения функций высших порядков
33. Обобщенные типы
34. Параметры обобщенных типов
35. Стирание и овеществление параметров типов
36. Обобщенные типы и подтипы
37. Объявление и применение аннотаций
38. Рефлексия: интроспекция объектов Kotlin во время выполнения
39. Понятие предметно-ориентированного языка
40. Внутренние предметно-ориентированные языки
41. Структура предметно-ориентированных языков
42. Создание разметки HTML с помощью внутреннего DSL
43. Создание структурированных API: лямбда-выражения с получателями в DSL
44. Гибкое вложение блоков с использованием соглашения «invoke»
45. Предметно-ориентированные языки Kotlin на практик
46. Сборка кода на Kotlin с помощью Gradle
47. Сборка проектов на Kotlin с помощью
48. Maven Сериализация JSON
49. Клиенты HTTP
50. Доступ к базам данных

Типовые формулировки задач

1. В файле приведён фрагмент базы данных «Продукты» о поставках товаров в магазины районов города. База данных состоит из трёх таблиц.

Таблица «Движение товаров» содержит записи о поставках товаров в магазины в течение первой декады июня 202 г., а также информацию о проданных товарах. Поле *Тип операции* содержит значение *Поступление* или *Продажа*, а в соответствующее поле *Количество упаковок, шт.* занесена информация о том, сколько упаковок товара поступило в магазин или было продано в течение дня. Заголовок таблицы имеет следующий вид.

ID операции	Дата	ID магазина	Артикул	Тип операции	Количество упаковок, шт.	Цена, руб./шт.

Таблица «Товар» содержит информацию об основных характеристиках каждого товара. Заголовок таблицы имеет следующий вид.

Артикул	Отдел	Наименование	Ед. изм.	Количество в упаковке	Поставщик

Таблица «Магазин» содержит информацию о местонахождении магазинов. Заголовок таблицы имеет следующий вид.

ID магазина	Район	Адрес

На рисунке приведена схема указанной базы данных.

Прочитайте информацию из Excel файла и определите на сколько увеличилось количество упаковок яиц диетических, имеющихся в наличии в магазинах Заречного района за период с 1 по 10 июня.

Для записи каждой таблицы построить класс, выбрать коллекцию для работы, решить задачу с помощью применения функциональных интерфейсов встроенных функций выбранной коллекции.

№ 2. Откройте файл электронной таблицы, содержащей в каждой строке пять натуральных чисел. Определите количество строк таблицы, в которых квадрат суммы максимального и минимального чисел в строке больше суммы квадратов трёх оставшихся.

№ 3. Построить обобщенный класс сортированное двоичное дерево. Построить конструктор класса, принимающий список. Построить метод, возвращающий отсортированный список. Построить методы, добавляющие элементы в дерево.

№ 4. Реализовать функцию, которая по трем спискам составляет список, состоящий из кортежей длины 3, где каждый кортеж (ai,bi,ci) с номером I получен следующим образом:

Ai – I по убыванию элемент первого списка

Bi – I по возрастанию суммы цифр элемент второго списка

Ci - I по убыванию количества делителей элемент третьего списка

Все элементы одного списка различны.

Если в списках B или C два элемента имеют одинаковый коэффициент, большим считается элемент, больший по абсолютной величине.

Решить данные задачи с применением возможностей класса list

№ 5. Дан файл, вывести в отдельный файл строки, состоящие из слов, не повторяющихся в исходном файле.

Логическое программирование на примере языка Swi-Prolog

1. Опишите понятие и структуру фактов в языке Пролог. Расскройте основные возможные типы, опишите понятие атом. Расскажите принцип работы терминальной машины Swi-Prolog, объясните каким образом происходит обработка вопросов.

2. Опишите смысл термина унификация, приведите показательные примеры. Объясните, как задаются предикаты, что такое правила и каким образом происходит работа с ними.

3. На примере числовых алгоритмов объясните смысл рекурсии вверх и рекурсии вниз в прологе.

4. Раскройте на примерах понятие backtracking, оператор отсечения и смысл его применения.

5. Опишите принцип работы со списками Черча в Swi-prolog. Покажите реализации встроенных предикатов работы со списками на основе механизма унификации(append, reverse nth0).

6. Объясните принцип работы со строками. Покажите на примерах основные встроенные предикаты работы со строками.

7. Покажите, каким образом происходит построение стандартных комбинаторных объектов средствами Swi-prolog.

8. Покажите основные принципы реализации переборных алгоритмов на графах средствами пролога.

9. Раскройте понятия статические и динамические факты. Поясните на примерах принцип работы с динамическими фактами.

10. Объясните принцип работы предикатов var, nonvar, atom, atomic, name, functor, arg, repeat.

Типовые формулировки задач

1. Построить предикат, позволяющий получить подмножество исходного множества. На его основе решить задачу компоновки рюкзака минимального веса. Дан рюкзак с заданным объемом V. Дан набор объектов [[3,4], [5,6], [7,8], [4,5]], где первое число объем объекта, второе число вес объекта. Сначала решить задачу нахождения подмножества объектов, сумма объемов которых равна объему рюкзака. Далее найти такое подмножество объектов, сумма объектов которых равна объему рюкзака, а сумма весов объекта минимальна возможна.

2. Построить предикат, позволяющий получить сочетание по k элементов из исходного списка. Далее построить предикат, позволяющий построить размещение без повторений по m элементов. На основе построенных предикатов построить все слова длины 10, в которых 3 буквы a, еще одна буква встречается 3 раза, остальные буквы не повторяются. В алфавите 6 букв [abcdef]. Рассчитать количество таких слов, проверить, совпадает ли количество слов в итоговом файле с рассчетным.

3. Написать предикат, который по заданному графу возвращает произвольное максимальное паросочетание. Модифицировать предикат так, чтобы он выявлял наибольшее паросочетание.

4. Дан файл, вывести в отдельный файл строки, состоящие из слов, не повторяющихся в исходном файле.

5. Номер 3797 обладает интересным свойством. Будучи простым, можно непрерывно удалять цифры слева направо и оставаться простыми на каждом этапе: 3797, 797, 97 и 7. Аналогично мы можем работать справа налево: 3797, 379, 37 и 3.

Найдите сумму простых чисел, меньших 1000000 которые можно обрезать слева направо и справа налево.

ПРИМЕЧАНИЕ. 2, 3, 5 и 7 не считаются усеченными простыми числами.

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

Методические рекомендации, определяющие процедуры оценивания на экзамене:

Процедура промежуточной аттестации проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

Итоговой формой контроля сформированности компетенций у обучающихся по дисциплине является зачет. Студенты обязаны сдать зачет в соответствии с расписанием и учебным планом.

ФОС промежуточной аттестации состоит из заданий и результатов текущего контроля.

Форма проведения зачета: письменно.

Преподавателю предоставляется право задавать студентам дополнительные вопросы по всей учебной программе дисциплины.

Результат сдачи зачета заносится преподавателем в экзаменационную ведомость и зачетную книжку.

Оценивание уровня освоения дисциплины основывается на качестве выполнения студентом заданий текущего контроля и проекта.

Критерии оценки:

Зачтено –

выполнено 60% лабораторных работ, 60% контрольных работ и выполнен проект,
или выполнено 60% лабораторных работ, 60% контрольных работ и дан ответ на два вопроса в билете,
или выполнено 100% лабораторных работ, 60% контрольных работ, дан ответ на один вопрос в билете,
или выполнено 60% лабораторных работ, 100% контрольных работ, дан ответ на один вопрос в билете,
или выполнено 100% лабораторных работ, 100% контрольных работ
или выполнено 30% лабораторных работ, 60% контрольных работ, выполнен проект, дан ответ на один вопрос в билете
или выполнено 60% лабораторных работ, 30% контрольных работ, выполнен проект, дан ответ на один вопрос в билете
или выполнено 30% лабораторных работ, 60% контрольных работ, дан ответ на три вопроса в билете
или выполнено 60% лабораторных работ, 30% контрольных работ, дан ответ на три вопроса в билете
или выполнено 30% лабораторных работ, 30% контрольных работ, выполнен проект, дан ответ на два вопроса в билете
или выполнено 60% контрольных работ, выполнен проект, дан ответ на два вопроса в билете
или выполнено 60% лабораторных работ, выполнен проект, дан ответ на два вопроса в билете
или выполнено 30% контрольных работ, выполнен проект, дан ответ на три вопроса в билете
или выполнено 30% лабораторных работ, выполнен проект, дан ответ на три вопроса в билете

Не зачтено –не выполнено ни одно из требований на оценку зачтено.

Методические рекомендации, определяющие процедуры оценивания лабораторных работ:

Процедура оценивания лабораторных работ проходит в соответствии с Положением о текущем контроле и промежуточной аттестации обучающихся ФГБОУ ВО «КубГУ».

По каждой лабораторной работе оформляется отчет. Отчеты сдаются на проверку руководителю в течение курса по мере их выполнения, и защищаются студентами в установленном порядке.

При защите отчета студенту могут быть заданы вопросы и дополнительные задания по сути лабораторной работы, в том числе из списка контрольных вопросов к данной лабораторной работе. При неудовлетворительной оценке знаний студента по теме данного отчета, студент возвращается к повторному изучению соответствующих материалов, после чего допускается к повторной защите. Неудовлетворительно выполненный отчет также возвращается на доработку.

Отчет должен содержать заголовок, тему лабораторной работы, цель, задание, индивидуальную тему, описание хода выполнения работы, необходимые прикладные материалы (схемы, макеты документов и т.п.), в соответствии с требованиями к содержанию, и выводы по работе.

Оценочные средства для инвалидов и лиц с ограниченными возможностями здоровья выбираются с учетом их индивидуальных психофизических особенностей.

– при необходимости инвалидам и лицам с ограниченными возможностями здоровья предоставляется дополнительное время для подготовки ответа на экзамене;

– при проведении процедуры оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья предусматривается использование технических средств, необходимых им в связи с их индивидуальными особенностями;

– при необходимости для обучающихся с ограниченными возможностями здоровья и инвалидов процедура оценивания результатов обучения по дисциплине может проводиться в несколько этапов.

Процедура оценивания результатов обучения инвалидов и лиц с ограниченными возможностями здоровья по дисциплине предусматривает предоставление информации в формах, адаптированных к ограничениям их здоровья и восприятия информации:

Для лиц с нарушениями зрения:

– в печатной форме увеличенным шрифтом,

– в форме электронного документа.

Для лиц с нарушениями слуха:

– в печатной форме,

– в форме электронного документа.

Для лиц с нарушениями опорно-двигательного аппарата:

– в печатной форме,

– в форме электронного документа.

Данный перечень может быть конкретизирован в зависимости от контингента обучающихся.

4.3. Методические указания по организации вычислительной инфраструктуры

1. Цель и задачи

Цель: Сформировать у студентов практические навыки развертывания и управления современной инфраструктурой для поддержки полного жизненного цикла коллективной разработки программного обеспечения.

Задачи:

Освоить принципы инфраструктуры как кода (IaC).

Получить опыт работы с системами контейнеризации и оркестрации.

Настроить pipelines непрерывной интеграции и поставки (CI/CD).

Организовать удаленную совместную работу над кодом с использованием Git.

Обеспечить базовый мониторинг и управление конфигурациями.

2. Рекомендуемая архитектура инфраструктуры

Для каждого проектного коллектива (3-5 человек) рекомендуется развернуть следующую схему:

text

[Локальные машины разработчиков]

|

| (Git push/pull)

v

[Удаленный Git-хостинг (GitLab / GitHub)]

|

| (Webhook trigger)

v

[CI/CD Сервер (GitLab Runner / GitHub Actions)]

|

| (Docker build/push)

v

[Контейнерный реестр (Docker Hub / GitLab Registry)]

|

| (Deploy command)

v

[Продакшен-сервер / Хостинг-площадка]

|

v

[Контейнеризованное приложение (Docker Compose)]

3. Технологический стек и инструменты (рекомендуемый)

Обязательный минимум:

Система контроля версий: Git

Удаленный хостинг Git-репозиториев: GitHub, GitLab или Bitbucket. Предпочтение — GitLab, так как он предоставляет встроенный CI/CD, Issue Tracker и Container Registry в одном продукте.

Контейнеризация: Docker

Оркестрация контейнеров (для проектов повышенной сложности): Docker Compose

Инфраструктура как код (IaC):

Базовый уровень: docker-compose.yml

Продвинутый уровень: Ansible playbooks для деплоя или Terraform для управления облачной инфраструктурой (если есть доступ к облачным кредитам).

CI/CD:

GitLab CI/CD (.gitlab-ci.yml) или GitHub Actions (.github/workflows/).

Pipeline должен включать стадии: build, test, deploy.

Опционально (для углубленного изучения):

Веб-сервер / Reverse Proxy: Nginx (для раздачи статики и проксирования запросов к приложению).

Мониторинг: Prometheus + Grafana (для сбора и визуализации метрик), или готовые облачные решения (Datadog, New Relic — по бесплатным тарифам).

Базы данных: Управляемая облачная БД (AWS RDS, DigitalOcean Managed Database) либо контейнеризованная версия (PostgreSQL/MySQL в Docker).

4. Практические задания и этапы внедрения

Этап 1: Базовая настройка репозитория и CI/CD

Создать группу/организацию на выбранном Git-хостинге.

Создать проект, настроить доступ для всех членов команды.

Настроить .gitignore, README.md, LICENSE.

Создать базовый Dockerfile для приложения.

Написать простейший .gitlab-ci.yml (или аналог для GitHub Actions), который выполняет сборку Docker-образа и запуск unit-тестов.

Этап 2: Контейнеризация и оркестрация

Определить все сервисы приложения (frontend, backend, БД, кэш).

Создать docker-compose.yml для локальной разработки и для продакшена.

Настроить переменные окружения для управления конфигурацией.

Этап 3: Настройка автоматического деплоя

Зарегистрировать/настроить сервер для деплоя (рекомендуется VPS от DigitalOcean, VK Cloud, Selectel и др. с помесячной оплатой).

Настроить в CI/CD pipeline автоматический деплой на сервер при пуше в ветку main/master.

Способ: Использование Docker Compose на удаленном сервере через ssh.

Альтернативный способ: Развертывание через Ansible-плейбук, запускаемый из CI/CD.

Этап 4: Мониторинг и документация

Настроить логирование приложения.

(Опционально) Развернуть стек мониторинга (Prometheus+Grafana) для сбора базовых метрик (CPU, RAM, нагрузка сети).

Задокументировать всю инфраструктуру: схему, инструкции по развертыванию, описание pipeline.

5. Критерии оценки инфраструктурной составляющей проекта

Инфраструктура является неотъемлемой частью проектной работы и оценивается в рамках общего критерия по проекту.

На «Зачтено» необходимо:

Работоспособность: Приложение должно быть доступно по URL (или на проверочном сервере) после автоматического деплоя.

Контейнеризация: Приложение и все его зависимости должны быть упакованы в Docker-контейнеры.

CI/CD: Настроен автоматический pipeline, который проходит как минимум стадии build и test. В Git-репозитории видна история запусков pipeline.

Документация: В README.md присутствуют четкие инструкции по запуску проекта локально и описание процесса деплоя.

Работа в команде: Вклад в инфраструктурную часть распределен между членами команды (видно по истории коммитов).

4.4. Методические указания по организации лабораторных работ

1. Цель и задачи

Цель: Сформировать у студентов устойчивые практические навыки работы в распределенной команде разработчиков с использованием современного инструментария и методологий.

Задачи:

Освоить жизненный цикл коллективной разработки от планирования до поставки ПО.

Сформировать компетенции работы с системами контроля версий.

Приобрести опыт настройки процессов непрерывной интеграции и поставки.
Отработать процедуры код-ревью и совместной разработки.
Закрепить навыки контейнеризации приложений.

2. Формат проведения работ
Периодичность: 6 лабораторных работ в течение семестра
Формат: Выполнение в командах по 3-4 человека
Сдача: Защита каждой работы с демонстрацией результата
Ведение: Общий репозиторий команды с историей выполнения всех ЛР

3. Структура и содержание лабораторных работ
Лабораторная работа 1: «Организация рабочего процесса команды»
Цель: Настроить инфраструктуру для коллективной работы и освоить базовые workflow Git.
Задачи:
Создать организацию/группу на GitHub/GitLab
Настроить SSH-ключи для всех участников
Создать проект с README, .gitignore, LICENSE

Отработать Git Flow: fork → clone → feature branch → commit → push → Pull Request
Настроить Issues и Projects для трекинга задач
Результат: Репозиторий с настроенным доступом для всех членов команды, минимум 3 merged PR от разных участников.

Лабораторная работа 2: «Проектирование и начало разработки»
Цель: Спроектировать архитектуру приложения и начать разработку по методологии Scrum.
Задачи:
Провести планирование спринта (2 недели)
Разработать техническое задание и архитектурную схему
Создать milestone и распределить задачи между участниками
Реализовать базовую структуру приложения
Провести код-ревью всех изменений
Результат: ТЗ в wiki, заполненный backlog, работающая заготовка приложения.

Лабораторная работа 3: «Настройка CI/CD пайплайна»
Цель: Настроить автоматическую сборку и тестирование проекта.
Задачи:
Написать Dockerfile для приложения
Настроить CI-пайплайн для автоматического запуска тестов
Добавить статический анализ кода (linters)
Настроить сборку Docker-образов
Добавить бейджи статуса сборки в README
Результат: Рабочий CI-пайплайн, проходящий все стадии, Docker-образ в registry.

Лабораторная работа 4: «Внедрение практик код-ревью»
Цель: Отработать процессы проверки кода и управления качеством.
Задачи:
Настроить protected branches
Создать checklist для код-ревью
Реализовать feature с обязательным ревью 2 участников

Настроить template для Pull Request
Внедрить требование успешной сборки для мержа PR

Результат: Процесс код-ревью документально оформлен и отработан на практике.

Лабораторная работа 5: «Настройка окружений и деплоя»
Цель: Настроить многоступенчатый деплой приложения.
Задачи:
Создать docker-compose.yml для разработки и продакшена
Настроить CD-пайплайн для деплоя на staging-сервер
Реализовать деплой по тегам на production-сервер
Настроить миграции БД при деплое
Добавить health-check для контейнеров
Результат: Приложение автоматически деплоится на два окружения.

Лабораторная работа 6: «Мониторинг и документация»
Цель: Настроить мониторинг работы приложения и завершить документацию.
Задачи:
Настроить логирование в контейнерах
Добавить сбор метрик приложения
Настроить дашборд для мониторинга
Задокументировать API (Swagger/OpenAPI)
Составить инструкцию по развертыванию и troubleshooting
Результат: Полная документация проекта, работающая система мониторинга.

4. Критерии оценки лабораторных работ
Общие требования для защиты:
Соблюдение сроков сдачи
Участие всех членов команды
Соответствие заданиям ЛР
Качество выполнения и оформления
Критерии оценки каждой ЛР:

Критерий	Вес	Описание
Техническая реализация	40%	Корректность выполнения, работоспособность решения
Работа в команде	30%	Вклад каждого участника, история коммитов, код-ревью
Документация	20%	Описание решения, скриншоты, инструкции
Качество кода	10%	Соблюдение code style, чистота репозитория

Шкала оценки:
«Зачтено»: 70-100% от максимального балла
«Не зачтено»: менее 70% от максимального балла

5. Методические рекомендации
Для студентов:
Распределяйте роли в команде (тимлид, разработчик, тестировщик, DevOps)
Проводите ежедневные стендапы (10-15 минут)
Ведите подробную документацию всех процессов

Своевременно обращайтесь за консультациями

6. Требования к отчетности
Для каждой ЛР команда предоставляет:
Ссылку на репозиторий с выполненной работой
Пулл-реквесты с выполненными задачами
Скриншоты работающего функционала
Краткий отчет в wiki репозитория:
Цель работы
Выполненные задачи
Распределение работы между участниками
Проблемы и пути их решения
Выводы

7. Рекомендуемый инструментарий
Хостинг кода: GitHub Education, GitLab
CI/CD: GitHub Actions, GitLab CI
Контейнеризация: Docker, Docker Compose
Коммуникация: Telegram, Discord, Mattermost
Документация: Wiki GitHub/GitLab, Markdown

4.5. Методические указания по организации проектной деятельности студентов

1. Цель и задачи

Цель: Интегрировать полученные в ходе лабораторных работ знания и навыки в рамках реализации сквозного проекта, максимально приближенного к реальным условиям промышленной разработки.

Задачи:

Сформировать опыт полного жизненного цикла разработки ПО в команде

Закрепить навыки проектного планирования, управления задачами и командной коммуникации

Развить умение принимать архитектурные и технологические решения

Отработать процедуры поставки и сопровождения программного продукта

2. Организация проектной деятельности
Продолжительность: 1 семестр (параллельно с лабораторными работами)
Формат: Сквозной проект с поэтапной сдачей результатов
Команды: 3-4 человека с распределением ролей
Процесс: Гибкая методология разработки (Scrum/Kanban)
Результат: Работающее приложение с полной инфраструктурой

3. Этапы выполнения проекта
Этап 1: Инициация и планирование (Недели 1-3)

Результат: Утвержденное техническое задание и план проекта
Содержание:
Выбор темы проекта из утвержденного перечня или предложение собственной
Формирование команды, распределение ролей (Team Lead, Developer, DevOps, QA)
Разработка технического задания с описанием функциональных требований
Создание дорожной карты (Roadmap) проекта
Планирование первого спринта, создание бэклога продукта
Настройка проектной инфраструктуры (репозиторий, Issue Tracker, CI/CD)

Этап 2: Проектирование архитектуры (Недели 4-5)
Результат: Архитектурная документация и прототип
Содержание:
Выбор технологического стека
Проектирование архитектуры системы (микросервисы/монолит)
Проектирование схемы базы данных
Разработка API-спецификации (OpenAPI/Swagger)
Создание прототипа основного функционала
Презентация архитектурного решения

Этап 3: Активная разработка (Недели 6-12)
Результат: Работающая система с основным функционалом
Содержание:
Проведение регулярных спринтов (2-3 недели каждый)
Ежедневные стендапы, планирование спринтов, ретроспективы
Реализация функциональности согласно бэклогу
Настройка и поддержка CI/CD пайплайна
Написание unit- и integration-тестов
Регулярное код-ревью и мерж Pull Requests
Деплой на тестовые окружения

Этап 4: Тестирование и стабилизация (Недели 13-14)
Результат: Стабильная версия продукта
Содержание:
Интеграционное и системное тестирование
Исправление критических ошибок
Нагрузочное тестирование (опционально)
Подготовка production-окружения
Финальный деплой и проверка безопасности

Этап 5: Защита проекта (Недели 15-16)
Результат: Презентация и демонстрация проекта
Содержание:
Подготовка финальной документации
Создание презентации проекта
Демонстрация работающего продукта
Ответы на вопросы комиссии

4. Роли в команде
Team Lead:
Координация работы команды
Ведение бэклога продукта
Проведение совещаний
Контроль сроков и качества
Developer (2 человека):
Реализация функциональности
Написание тестов
Участие в код-ревью
Рефакторинг кода
DevOps Engineer:
Настройка инфраструктуры
Поддержка CI/CD

Деплой и мониторинг
Контейнеризация приложения

5. Критерии оценки проекта
Общий вес проекта в итоговой оценке: 50%

Критерий	Вес	Показатели оценки
Техническая реализация	30%	Работоспособность, соответствие ТЗ, качество кода, архитектурные решения
Процесс разработки	25%	Следование методологии, история коммитов, код-ревью, управление задачами
Инфраструктура и DevOps	20%	CI/CD пайплайн, контейнеризация, автоматизация деплоя, мониторинг
Документация	15%	Техническое задание, архитектурная документация, руководство пользователя
Презентация и защита	10%	Качество демонстрации, ответы на вопросы, командная работа

Минимальные требования для защиты:
Реализован основной заявленный функционал
Проект запускается по инструкции и работает
Настроен CI/CD пайплайн
Все члены команды внесли вклад в разработку
Предоставлена полная документация

6. Требования к проекту
Обязательные компоненты:
Система контроля версий (Git) с историей разработки
CI/CD пайплайн с автоматическими тестами
Контейнеризация приложения (Docker)
Автоматический деплой на сервер
Документация в README.md и wiki
Issue tracking с историей задач

5. Перечень учебной литературы, информационных ресурсов и технологий

5.1. Учебная литература

1. Рубио-Санчес, М. Введение в рекурсивное программирование : учебное пособие : [16+] / М. Рубио-Санчес ; пер. с англ. Е. В. Борисова. – Москва : ДМК Пресс, 2019. – 437 с. : ил. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=596896 (дата обращения: 29.05.2024). – ISBN 978-5-97060-703-9. – Текст : электронный.

2. Жемеров, Д. Kotlin в действии : практическое пособие : [16+] / Д. Жемеров, С. Исакова. – Москва : ДМК Пресс, 2018. – 402 с. : ил., табл. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=578703 (дата обращения: 29.05.2024). – ISBN 978-5-97060-497-7. – Текст : электронный.

3. Цуканова, Н. И. Технология разработки экспертных систем на языке Visual Prolog 7.5 : учебное пособие / Н. И. Цуканова, К. А. Майков. – Москва : Курс, [2023]. – 250 с. : ил., табл. – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=708089 (дата обращения: 29.05.2024). – Библиогр.: с. 205-206. – ISBN 978-5-906923-40-0. – Текст : электронный

4. Боровская, Е. В. Основы искусственного интеллекта : учебное пособие : [16+] / Е. В. Боровская, Н. А. Давыдова. – 4-е изд., электрон. – Москва : Лаборатория знаний, 2020. – 130 с. : схем. – (Педагогическое образование). – Режим доступа: по подписке. – URL: https://biblioclub.ru/index.php?page=book&id=595419 (дата обращения: 29.05.2024). – Библиогр. в кн. – ISBN 978-5-00101-908-4. – Текст : электронный

5.Клоксин, Уильям. Программирование на языке Пролог : / У. Клоксин, К. Меллиш ; пер.: А. В. Горбунов, М. М. Комаров ; ред. пер.: А. К. Платонов, Ю. М. Лазутин. - М. : Мир, 1987. - 336 с. : ил. - (Математическое обеспечение ЭВМ). - Предм. указ.: с. 335-336.

6. Большакова Елена Игоревна, Груздева Надежда Валерьевна Программирование на языке Пролог: Учебное пособие. – М.: Издательский отдел факультета ВМК МГУ имени М.В.Ломоносова

(лицензия ИД № 05899 от 24.09.2001); МАКС Пресс, 2013 – 112 с.

7. Филд А... Функциональное программирование. (Functional Programming) [Djv-13.6M] Авторы: А. Филд, П. Харрисон. Перевод с английского М.В. Горбатовой, А.А. Рябинина, В.Л. Торхова, М.В. Федорова под редакцией В.А. Горбатова. Научное издание. (Москва: Издательство «Мир». Редакция литературы по информатике, 1993)

8. John Harrison, Introduction to Functional Programming http://www.cl.cam.ac.uk/teaching/Lectures/funprog-jrh-1996/ (русский перевод: http://code.google.com/p/funprog-ru/)

9. Малышев Д.С., Мокеев Д.Б. Некоторые элементы неклассических логик и лямбда-исчисления: учебно-методическое пособие. — [электронный ресурс] — Нижний Новгород: Нижегородский госуниверситет, 2017 — 32 с.

10. Официальный сайт языка Котлин https://kotlinlang.org/

11. Руководство по языку Kotlin https://metanit.com/kotlin/tutorial/

12. Руководство по языку Kotlin https://kotlinlang.ru/

13. Гриффитс Дон, Гриффитс Дэвид Head First. Kotlin. — СПб.: Питер, 2020. — 464 с.: ил. — (Серия «Head First O'Reilly»). ISBN 978-5-4461-1335-4.

5.2. Периодические издания:

1 Базы данных компании «Ист Вью» http://dlib.eastview.com
2 Электронная библиотека GREBENNIKON.RU https://grebennikon.ru/

5.3. Интернет-ресурсы, в том числе современные профессиональные базы данных и информационные справочные системы

Электронно-библиотечные системы (ЭБС):

1 ЭБС «ЮРАЙТ» https://urait.ru/
2 ЭБС «УНИВЕРСИТЕТСКАЯ БИБЛИОТЕКА ОНЛАЙН» http://www.biblioclub.ru/
3 ЭБС «BOOK.ru» https://www.book.ru
4 ЭБС «ZNANIUM.COM» www.znanium.com
5 ЭБС «ЛАНЬ» https://e.lanbook.com

Профессиональные базы данных

1 Scopus http://www.scopus.com/
2 ScienceDirect https://www.sciencedirect.com/
3 Журналы издательства Wiley https://onlinelibrary.wiley.com/
4 Научная электронная библиотека (НЭБ) http://www.elibrary.ru/
5 Полнотекстовые архивы ведущих западных научных журналов на Российской платформе научных журналов НЭИКОН http://archive.neicon.ru
6 Национальная электронная библиотека (доступ к Электронной библиотеке диссертаций Российской государственной библиотеки (РГБ) https://rusneb.ru/
7 Президентская библиотека им. Б.Н. Ельцина https://www.prlib.ru/

8 База данных CSD Кембриджского центра кристаллографических данных (CCDC) https://www.ccdc.cam.ac.uk/structures/
9 Springer Journals: https://link.springer.com/
10 Springer Journals Archive: https://link.springer.com/
11 Nature Journals: https://www.nature.com/
12 Springer Nature Protocols and Methods: https://experiments.springernature.com/sources/springer-protocols
13 Springer Materials: http://materials.springer.com/
14 Nano Database: https://nano.nature.com/
15 Springer eBooks (i.e. 2020 eBook collections): https://link.springer.com/
16 "Лекториум ТВ" http://www.lektorium.tv/
17 Университетская информационная система РОССИЯ http://uisrussia.msu.ru

Бесплатные образовательные ресурсы

1 Jupyter Notebook – интерактивные вычисления
2 Visual Studio Code – редактор кода с поддержкой Python
3 Google Scholar/arXiv – доступ к научным публикациям

Ресурсы свободного доступа

1 КиберЛенинка http://cyberleninka.ru/;
2 Американская патентная база данных http://www.uspto.gov/patft/
3 Министерство науки и высшего образования Российской Федерации https://www.minobrnauki.gov.ru/;
4 Федеральный портал "Российское образование" http://www.edu.ru/;
5 Информационная система "Единое окно доступа к образовательным ресурсам" http://window.edu.ru/;
6 Единая коллекция цифровых образовательных ресурсов http://school-collection.edu.ru/ .
7 Проект Государственного института русского языка имени А.С. Пушкина "Образование на русском" https://pushkininstitute.ru/;
8 Справочно-информационный портал "Русский язык" http://gramota.ru/;
9 Служба тематических толковых словарей http://www.glossary.ru/;
10 Словари и энциклопедии http://dic.academic.ru/;
11 Образовательный портал "Учеба" http://www.ucheba.com/;
12 Законопроект "Об образовании в Российской Федерации". Вопросы и ответы http://xn--273--84d1f.xn--p1ai/voprosy_i_otvety

Собственные электронные образовательные и информационные ресурсы КубГУ

1 Электронный каталог Научной библиотеки КубГУ http://megapro.kubsu.ru/MegaPro/Web
2 Электронная библиотека трудов ученых КубГУ http://megapro.kubsu.ru/MegaPro/UserEntry?Action=ToDb&idb=6
3 Среда модульного динамического обучения http://moodle.kubsu.ru
4 База учебных планов, учебно-методических комплексов, публикаций и конференций http://infoneeds.kubsu.ru/
5 Библиотека информационных ресурсов кафедры информационных образовательных технологий http://mschool.kubsu.ru;
6 Электронный архив документов КубГУ http://docspace.kubsu.ru/
7 Электронные образовательные ресурсы кафедры информационных систем и технологий в образовании КубГУ и научно-методического журнала "ШКОЛЬНЫЕ ГОДЫ" http://icdau.kubsu.ru/

5.4 Перечень информационно-коммуникационных технологий

1. Облачные платформы и сервисы
cloud.ru,YandexCloud,AWS/GCP/Azure– облачные вычисления

2. Системы управления версиями и коллаборации
Git/GitHub/GitLab – контроль версий кода и совместная разработка

4. Система управления обучением
Moodle – сдача работ

5.5 Перечень лицензионного и свободно распространяемого программного обеспечения

1. Свободное ПО (Open Source)
GitLab, GIT, MLFlow, Docker, Kubernetes, Terraform.

6. Методические указания для обучающихся по освоению дисциплины (модуля)

По курсу предусмотрено проведение лекционных занятий, на которых дается систематизированный материал по логическому программированию, лямбда исчислению и функциональному программированию средствами Котлин. В ходе лекций рассматриваются ключевые концепции. После каждой лекции рекомендуется выполнение практических заданий для закрепления ключевых понятий и методов.

Лабораторные занятия курса посвящены разработке в логической и функциональной парадигмах и решению задач по лямбда-исчислению.

При самостоятельной работе студентам необходимо изучать рекомендованную литературу в виде официальной документации к используемым открытым программным продуктам, облачным платформам.

Для студентов с ограниченными возможностями здоровья предусмотрены дополнительные индивидуальные консультации, на которых преподаватель подробно разъясняет сложные аспекты дисциплины, помогает адаптировать практические задания и обеспечивает специальные условия для освоения методов работы с системами искусственного интеллекта. Индивидуальный подход позволяет таким студентам полноценно участвовать в учебном процессе и достигать требуемых результатов обучения.

7. Материально-техническое обеспечение по дисциплине (модулю)

Виртуальные машины, кластер Managed Kubernetes и ресурсы GPU в облаке предоставляется индустриальным партнером ПАО «Сбербанк»:

№	Продукт	Параметры продукта	Кол-во	Кол-во конфигураций	Ед. изм.
1	Виртуальная машина	Виртуальная машина 10% vCPU 2 vCPU 4 RAM	1	60	Шт
		ОС Ubuntu 22.04	1		Шт
		Системный диск SSD	1		Шт

Дополнительные облачные ресурсы предоставляются технологическим партнером Yandex Cloud.

№	Вид работ	Наименование учебной аудитории, ее оснащенность оборудованием и техническими средствами обучения
1	Лекционные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения
2	Лабораторные занятия	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, проектором, программным обеспечением
3	Групповые (индивидуальные) консультации	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
4	Текущий контроль, промежуточная аттестация	Аудитория, укомплектованная специализированной мебелью и техническими средствами обучения, компьютерами, программным обеспечением
5	Самостоятельная работа	Кабинет для самостоятельной работы, оснащенный компьютерной техникой с возможностью подключения к сети «Интернет», программой экранного увеличения и обеспеченный доступом в электронную информационно-образовательную среду университета.

Примечание: Конкретизация аудиторий и их оснащение определяется ОПОП.