ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«25» мая 2026 г.
Протокол № 2

Рабочая программа дисциплины (модуля)
«Функциональное программирование»

Направление подготовки: *02.04.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *ИИ в биотехе*

Специализация: *Структурная биоинформатика*

Квалификация (степень) выпускника: *магистр*

Форма обучения: *очная*

Срок освоения программы: *2 года*

Год набора: *2026*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

Сертификат: 0215B0C000D1B3FEB44A37EE631E0C27D4
Владелец: Ивашкевич Евгений Васильевич
Действителен: с 13.01.2026 до 13.04.2027

Электронный документ

Москва
2026

Содержание

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Функциональное программирование» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – магистратура по направлению 02.04.01 Математика и компьютерные науки, профиль ИИ в биотехе, утвержденный приказом Министерства науки и высшего образования Российской Федерации № 810 от 23.08.2017 года.

Изучение дисциплины (модуля) «Функциональное программирование» формирует у студентов способность создавать надёжное, модульное и математически обоснованное программное обеспечение, что делает их конкурентоспособными в областях, требующих высокой степени типовой безопасности, корректности и сложной абстракции, таких как разработка компиляторов, блокчейн-технологии, финансовые системы и научные вычисления.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки магистратуры по направлению 02.04.01 Математика и компьютерные науки, профиль ИИ в биотехе и входит в обязательную часть Блока 1 как дисциплина по выбору специализации «Структурная биоинформатика».

Дисциплина (модуль) изучается на 2 курсе в 4 семестре.

Цель изучения дисциплины (модуля): формирование у обучающихся глубокого понимания теоретических основ и практических принципов функционального программирования на основе языка Haskell, развитие навыков типобезопасной, модульной и математически обоснованной разработки программного обеспечения, а также выработка способности применять абстракции теории категорий и универсальной алгебры в реальных задачах программирования.

Задачи изучения дисциплины (модуля):

- освоить синтаксис и семантику Haskell, включая полиморфизм, ленивые вычисления, рекурсию и системы типов;
- изучить иерархию классов типов (полугруппы, моноиды, функторы, аппликативы, монады) и их математические законы;
- научиться проектировать программы с использованием продвинутых техник: GADT, семейств типов, монадные трансформеры, tagless final, Free-монады;
- овладеть методами доказательства корректности программ по индукции и обеспечения типобезопасности на этапе компиляции;
- развить навыки разработки, тестирования, профилирования и развертывания функциональных программ с использованием промышленных инструментов (Cabal, Stack, property-based testing, STM, streamly).

В результате освоения дисциплины (модуля) обучающийся должен:

знать:

- основы теории категорий, универсальной алгебры и дискретной математики: объекты, морфизмы, функторы, декартово замкнутые категории, моноидальные категории; связь с конструкциями Haskell (функторы, аппликативы, монады, линзы, дифференцирование);
- промышленные стандарты разработки на Haskell: система модулей, инструменты сборки (cabal/stack), лучшие практики организации кода (best practices), работа с

эффектами, тестирование, профилирование;

— синтаксис и базовые конструкции Haskell: полиморфизм, функции высшего порядка, ленивые вычисления, рекурсия, доказательства по индукции для программ на Haskell;

— иерархию стандартных классов типов: полугруппы, моноиды, функторы, аппликативные функторы, монады (IO, ST), Traversable, Foldable; их законы и взаимосвязи;

— продвинутые техники работы с типами: GADT (интерпретатор STLC, типобезопасные преобразования), семейства типов, уточняющие типы (refined), доказательства с constraint;

— концепции монадных трансформеров, эффектов и конкурентного/реактивного программирования: композиция аппликативных функторов, трансформеры монад, tagless final, effectful, Free-монада, Term-монада, STM, многопоточность, reactive-banana.

уметь:

— решать несложные алгоритмические задачи в функциональной парадигме и доказывать корректность решения: писать рекурсивные и полиморфные функции с использованием списков, алгебраических типов данных, классов типов (Functor, Applicative, Monad); осознанно применять ленивые вычисления;

— строить парсеры с помощью аппликативных комбинаторов; использовать do-нотацию для IO и генераторных выражений; писать генераторы случайных данных (ГПСЧ) и тесты на основе свойств (property-based testing);

— проектировать программы с использованием трансформеров монад (ReaderT + ExceptT + IO) или tagless final; применять библиотеку streamly для потоковой обработки;

— реализовывать интерпретаторы маленьких языков с помощью GADT и Free-монады; использовать GADT для типобезопасных преобразований (безопасные векторы, SQL-запросы через rel8);

— самостоятельно реализовывать небольшие проекты на Haskell: применять линзы (вывод Лаарховена) для работы с вложенными структурами; писать многопоточные программы с STM; создавать простые веб-серверы на scotty.

владеть:

— функциональным подходом к решению задач программирования и математики: навык функционального мышления и рассуждения о программах; проведение доказательств свойств функций по индукции; понимание множества значений типа и комбинаторики на типах (биекции);

— инструментарием композиции эффектов: выбор между монадными трансформерами, tagless final и effectful в зависимости от задачи; настройка эффектов (исключения, ресурсы, чтение/запись состояния);

— техниками обеспечения типобезопасности на уровне компиляции: использование GADT, семейств типов, refined-типов и доказательств с constraint для исключения ошибок времени выполнения (непустые списки, корректность SQL-запросов через rel8);

— методами конкурентного и реактивного программирования: отладка livelock в STM; построение FRP-систем на reactive-banana; использование Weak и StableName для

работы с указателями и предотвращения утечек;

— пониманием теоретических основ (теория категорий, универсальная алгебра) в их практическом применении к разработке на Haskell.

2. Перечень планируемых результатов обучения

Компетенции, формируемые в результате освоения дисциплины (модуля) при проведении учебных занятий в форме контактной работы обучающихся с педагогическими работниками Университета и в форме самостоятельной работы обучающихся:

Компетенция	Содержание компетенции	Индикатор компетенции	Перечень планируемых результатов обучения по дисциплине (модулю)
УК-1.	Способен осуществлять критический анализ проблемных ситуаций на основе системного подхода, вырабатывать стратегию действий	УК-1.1.	Знает основные принципы системного подхода и методы критического анализа, а также теоретические основы стратегического планирования и принятия решений
		УК-1.2.	Умеет применять методы системного анализа для выявления ключевых проблемных ситуаций, формулировать гипотезы и разрабатывать альтернативные стратегии действий на основе полученных данных
		УК-1.3.	Имеет практический опыт в проведении анализа реальных проблемных ситуаций в рамках проектов, способен вырабатывать и обосновывать стратегии действий, учитывая различные аспекты и последствия
УК-2.	Способен управлять проектом на всех этапах его жизненного цикла	УК-2.1.	Знает основные методологии управления проектами, ключевые этапы жизненного цикла проекта и инструменты для планирования и контроля
		УК-2.2.	Умеет разрабатывать проектную документацию, устанавливать цели и задачи проекта, а также эффективно распределять ресурсы и управлять рисками на всех этапах его реализации
		УК-2.3.	Имеет практический опыт в управлении реальными проектами, включая планирование, исполнение и завершение, а также в оценке результатов и проведении анализа успешности проекта
ПК-1.	Способен определять общие формы и закономерности области искусственного	ПК-1.1.	Знает основные теоретические концепции и принципы искусственного интеллекта в биотехнологии, а также методы

	интеллекта в биологии, медицине или биотехнологии, обрабатывать и интерпретировать данные биологических исследований		обработки, анализа и визуализации данных в биологии, медицине или биотехнологии
		ПК-1.2.	Умеет проводить систематический анализ области профессиональной деятельности, выявлять закономерности и тенденции, моделировать основные биологические процессы и интерпретировать полученные результаты с использованием методов биоинформатики и искусственного интеллекта
		ПК-1.3.	Имеет практический опыт работы в области искусственного интеллекта в биотехнологии, включая участие в научных проектах, исследованиях или практических заданиях
ПК-2.	Способен математически корректно ставить и решать естественнонаучные и прикладные задачи в области в области биологии, медицины или биотехнологий	ПК-2.1.	Знает основные методы математического моделирования и теоретические основы современной биологии и биоинформатики, необходимые для корректной постановки задач
		ПК-2.2.	Умеет формулировать математические модели биологических процессов, выбирать адекватные методы решения и интерпретировать результаты в профессиональном контексте
		ПК-2.3.	Имеет практический опыт постановки и решения прикладных задач с использованием методов ИИ в области биологии, медицины или биотехнологий
ПК-3.	Способен решать задачи профессиональной деятельности и применять методы искусственного интеллекта в прикладных и фундаментальных исследованиях в области биологии, медицины или биотехнологий	ПК-3.1.	Знает принципы и методы решения профессиональных задач, включая применение профильно-специализированных знаний в области ИИ для фундаментальных и прикладных исследований в области биологии, медицины или биотехнологий
		ПК-3.2.	Умеет применять математические и компьютерные методы, формулировать результаты и анализировать их последствия для научных и практических решений

		ПК-3.3.	Имеет опыт участия в проектах, где методы ИИ применялись для решения исследовательских и прикладных задач в области биологии, медицины или биотехнологий
ПК-5.	Способен передавать результаты прикладных задач в виде профессиональных рекомендаций в области искусственного интеллекта, биоинформатики и биотехнологии	ПК-5.1.	Знает методы формулирования рекомендаций и терминологию области искусственного интеллекта и биоинформатики
		ПК-5.2.	Умеет интерпретировать результаты анализа и моделирования и формулировать рекомендации для практического применения в области биологии, медицины или биотехнологий
		ПК-5.3.	Имеет опыт разработки рекомендаций по результатам проектов в области искусственного интеллекта и биотехнологии

3. Тематический план

№ п/п	Наименование раздела дисциплины (модуля)	Трудоемкость, академические часы				ТКУ (текущий контроль успеваемости)
		Очная форма				
		Аудиторная работа		Контроль	Самостоятельная работа	
		Лекции	Семинары (практические занятия)			
1	Введение в программирование на Haskell: от синтаксиса до монад	14	14	2	21	Домашние задания Проект
2	Продвинутое функциональное программирование	16	16	2	23	Домашние задания Проект
	Зачет с оценкой			***6***		Коллоквиум
	Итого:	***30***	***30***	***10***	***44***	
Объем дисциплины (модуля) (в ак. ч.)		***114***				
Объем дисциплины (модуля) (в зач. ед.)		***3***				

4. Содержание дисциплины (модуля)

№п/п	Наименование раздела дисциплины (модуля)	Содержание дисциплины (модуля) по темам
1	Введение в программирование на Haskell: от синтаксиса до монад	Синтаксис Haskell. Полиморфизм и функции высшего порядка. Списки. Рекурсия и ленивые вычисления. Доказательства по индукции. Алгебраические типы данных. Простые классы типов. Множество значений типа. Комбинаторика на типах. Биекции. (Стандартные) структуры данных. Полугруппы и моноиды. Функторы и свёртки. Решётки. Фундированные множества. Аппликативные функторы. Комбинаторы парсеров. Класс Traversable. Свёртки и стриминг. do-нотация для IO. Генераторные выражения. ГПСЧ. Класс Monad. Монада ST. Тестирование, основанное на свойствах (property-based testing). Многопоточность в Haskell. Программная транзакционная память (Software Transactional Memory (STM)). Веб-серверы на scotty. Живая блокировка (Livelock) и stm-containers. Композиция (аппликативных) функторов. Трансформеры монад. Бестеговый конечный стиль (tagless final). Стратегии дерайвинга. Библиотека streamly. Многочлены и булевы функции. Подстановка. Монада Free. Монада Term альфа-эквивалентных термов. Архитектура компиляторов.
2	Продвинутое функциональное программирование	Семейства типов. Системы эффектов на примере библиотеки effectful. Драйверы баз данных на примере rel8. Исключения и ресурсы. Уточняющие типы на примере библиотеки refined. Доказательства с помощью constraint. Введение в теорию категорий.

		GADT: интерпретатор STLC и другие примеры. Типобезопасные преобразования. Декартово замкнутые категории. Наивное определение линз. Вывод линз Лаарховена. Моноидальные категории. Игры, дифференцирование и линзы. Функциональное реактивное программирование на примере reactive-banana. Слабая ссылка (Weak) и стабильное имя (StableName). FFI (внешний интерфейс).

5. Учебно-методическое обеспечение

Университет располагает полным набором лицензионного и свободно распространяемого программного обеспечения, включая продукты отечественного производства.

Каждый студент в течение всего периода обучения получает индивидуальный неограниченный доступ к электронно-библиотечной системе и электронной информационно-образовательной среде университета. Эти системы предоставляют возможность доступа к ресурсам из любой точки, где есть подключение к сети Интернет, как на территории университета, так и за его пределами.

Студентам обеспечен удаленный доступ к современным профессиональным базам данных и информационным справочным системам.

Основная литература:

1. Липовача, М. Изучай Haskell во имя добра! : практическое руководство / М. Липовача ; пер. с англ. Д. Леушина, А. Синицына, Я. Арсанукаева. - 2-е изд. - Москва : ДМК Пресс, 2023. - 492 с. - ISBN 978-5-89818-338-7. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2102625.

2. Берд, Р. Жемчужины проектирования алгоритмов: функциональный подход. С примерами на языке Haskell : практическое руководство / Р. Берд ; пер. с англ. В. Н. Брагилевского, А. М. Пеленицына. - 2-е изд. - Москва : ДМК Пресс, 2023. - 331 с. - ISBN 978-5-89818-555-8. - Текст : электронный. - URL: https://znanium.com/catalog/product/2107906.

Дополнительная литература:

1. Окасаки, К. Чисто функциональные структуры данных : практическое руководство / К. Окасаки ; пер. с англ. Г. К. Бронникова ; под ред. В. Н. Брагилевского. - 2-е изд. - Москва : ДМК Пресс, 2023. - 252 с. - ISBN 978-5-89818-577-0. - Текст : электронный. - URL: https://znanium.com/catalog/product/2107940.

2. Лейно М., Р. Доказательство корректности программ : учебное пособие / К. Рустан М. Лейно ; пер. с англ. А. Н. Киселева. – Москва : ДМК Пресс, 2024. - 532 с. – ISBN 978-5-93700-199-3. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2204231.

6. Материально-техническое обеспечение

Университет располагает материально-технической базой, соответствующей действующим противопожарным правилам и нормам и обеспечивающей проведение всех видов дисциплинарной и междисциплинарной подготовки, практической и научно-исследовательской работ обучающихся, предусмотренных учебным планом.

Помещения, которые представляют собой учебные аудитории для проведения занятий лекционного типа, занятий семинарского (практического) типа, групповых и индивидуальных консультаций, текущего контроля и промежуточной аттестации, а также помещения для самостоятельной работы и помещения для хранения и профилактического обслуживания учебного оборудования. Помещения укомплектованы специализированной мебелью и техническими средствами обучения, служащими для представления учебной информации большой аудитории.

Изучение дисциплины (модуля) обеспечивается в учебных аудиториях, оснащенных:

— столами и стульями;

— компьютерной техникой;

— механическими калькуляторами;

— специализированным оборудованием, включая демонстрационное оборудование.

Помещения для самостоятельной работы обучающихся, в том числе приспособленные для использования инвалидами и лицами с ограниченными возможностями здоровья, оснащены компьютерной техникой с возможностью подключения к сети «Интернет» и обеспечением доступа в электронную информационно-образовательную среду Университета.

Обучающимся предоставляется доступ (в том числе удаленный) к ресурсам информационно-телекоммуникационной сети «Интернет», электронным ресурсам (в том числе электронным библиотечным системам, современным профессиональным базам данных и информационным справочным системам):

№	Наименование портала (издания, курса, документа)	Ссылка
1.	Научная электронная библиотека elibrary.ru библиотека	https://elibrary.ru/defaultx.asp
2.	База данных для IT-специалистов	https://habr.com
3.	База данных ScienceDirect	https://www.sciencedirect.com
4.	Официальный сайт Министерства науки и высшего образования Российской Федерации	https://minobrnauki.gov.ru/
5.	Федеральный портал «Российское образование"	https://www.edu.ru/
6.	Информационная система "Единое окно доступа к образовательным ресурсам"	http://window.edu.ru/
7.	Единая коллекция цифровых образовательных ресурсов	http://school-collection.edu.ru/
8.	Федеральный центр информационно - образовательных ресурсов	http://fcior.edu.ru/

Перечень информационных технологий, используемых при осуществлении образовательного процесса по дисциплине (модулю), в том числе комплект лицензионного программного обеспечения, современные профессиональные базы данных и информационные справочные системы:

Наименование ПО	Производство	Лицензионное / свободно распространяемое
Операционные системы:		
Microsoft Imagine (Windows Client, Server)	зарубежное	лицензионное
Браузеры:		
Яндекс.Браузер	отечественное	свободно распространяемое
Google Chrome	зарубежное	свободно распространяемое
Офисные приложения:		
Microsoft Imagine (Visio, OneNote)	зарубежное	лицензионное
TeXstudio	зарубежное	свободно распространяемое
Adobe Acrobat Reader	зарубежное	свободно распространяемое

Программное обеспечение для планирования и учета времени:		
Toggle app	зарубежное	свободно распространяемое
Системы управления проектами:		
Microsoft Imagine (Project)	зарубежное	лицензионное
Системы управления базами данных:		
Microsoft Imagine (SQL Server)	зарубежное	лицензионное
Системы резервного копирования (backup):		
Acronis Backup Advanced for HyperV	зарубежное	лицензионное
Справочно-правовые системы:		
КонсультантПлюс: справочно-правовая система	отечественное	лицензионное
Средства антивирусной защиты:		
Kaspersky Endpoint Security для бизнеса Стандартный Russian Edition	отечественное	лицензионное
Среды разработки:		
Visual Studio Code	зарубежное	свободно распространяемое
Bash (Unix shell)	зарубежное	свободно распространяемое
Anaconda	зарубежное	свободно распространяемое
Robotic Operating System	зарубежное	свободно распространяемое
CopelliaSim	зарубежное	свободно распространяемое
Google Colaboratory	зарубежное	свободно распространяемое
Пакеты программных средств и библиотек:		
AutoPsy	зарубежное	свободно распространяемое
Interactive Disassembler (IDA)	зарубежное	свободно распространяемое
Системы управления библиографической информацией:		
Zotero	зарубежное	свободно распространяемое
Сервисы и службы:		
Bind	зарубежное	свободно распространяемое
Docker	зарубежное	свободно распространяемое

7. Методические и оценочные материалы

Методические указания для обучающихся по освоению дисциплины (модуля)

В процессе изучения дисциплины (модуля) «Функциональное программирование» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, домашние задания, проект, коллоквиум, а также различные виды самостоятельной работы обучающихся по заданию преподавателя, направленные на развитие навыков профессиональной лексики, закрепление практических профессиональных компетенций, поощрение инициатив.

Лекция – систематическое, последовательное, монологическое изложение преподавателем учебного материала, как правило, теоретического характера.

В процессе лекций рекомендуется вести конспект лекций: кратко и схематично фиксировать основные идеи, выводы и обобщения лекции; выделять важные мысли, ключевые слова и термины. Необходимо отметить вопросы или материалы, которые вызывают затруднения, и попытаться найти ответы в рекомендованной литературе. Если

разобраться в материале не удается, следует сформулировать вопрос и задать его преподавателю на консультации или во время семинарского (практического) занятия.

Семинар – это форма учебной деятельности, проводимая в учебном заведении под руководством преподавателя, где студенты активно участвуют в обсуждениях, практических заданиях и других формах взаимодействия.

Для успешной подготовки к семинару рекомендуется заранее ознакомиться с темой занятия и основными материалами, чтобы иметь возможность активно участвовать в обсуждении. Также полезно подготовить вопросы и идеи для обсуждения, что поможет глубже понять материал и продемонстрировать заинтересованность.

Домашнее задание – набор задач по темам недели.

При работе над домашними заданиями важно внимательно ознакомиться с требованиями и сроками выполнения. Рекомендуется разбивать задания на этапы, чтобы избежать перегрузки и лучше усвоить материал, использовать различные источники информации, включая учебники и онлайн-ресурсы, для более глубокого понимания темы.

Проект – исследовательская работа по дисциплине (модулю) и презентация результатов.

Для успешной подготовки к проекту рекомендуется: четко определить цели и задачи проекта; составить план работы, разбив проект на этапы с указанием сроков выполнения каждого из них; использовать разнообразные источники информации и инструменты для исследования темы; регулярно проверять прогресс и вносить коррективы в план, если это необходимо.

Коллоквиум – устные ответы на вопросы, список которых известен студенту заранее.

В процессе подготовки к коллоквиуму необходимо проанализировать учебные материалы, ознакомившись с лекциями, учебниками и дополнительными источниками, акцентируя внимание на ключевых темах. Рекомендуется создать структурированные конспекты, выделяя основные идеи, термины и формулы.

Бонусные баллы — это оценки, которые студенты могут получить за выполнение дополнительных заданий.

Формат бонусных баллов позволяет студентам улучшить общую оценку по дисциплине (модулю) и стимулирует углубленное изучение материала.

Самостоятельная работа – работа студентов, направленная на углубленное изучение отдельных тем и вопросов учебной дисциплины (модуля).

В процессе самостоятельной работы студенты взаимодействуют с рекомендованными материалами при минимальном участии преподавателя. Задачи студента включают работу с конспектами лекций (обработка текста), повторное изучение учебных материалов планов и тезисов ответов, изучение дополнительных тем, выполнение учебно-исследовательских заданий и другое.

Система оценивания результатов обучения по дисциплине (модулю)

Критерии получения уровня и оценивания сформированности компетенций по дисциплине (модулю) «Функциональное программирование»

Оценивание уровня учебных достижений обучающихся по дисциплине (модулю) осуществляется в виде текущего контроля успеваемости и промежуточной аттестации.

Промежуточная аттестация по дисциплине (модулю) осуществляется в форме ***зачета с оценкой***, при этом проводится оценка компетенций, сформированных по дисциплине.

Для оценивания текущего контроля успеваемости и промежуточной аттестации используется десятибалльная шкала оценивания, которая соотносится с традиционной пятибалльной шкалой следующим образом, если иное не указано в силлабусе дисциплины:

Десятибалльная оценка	**Пятибалльная оценка**	**Общая характеристика результата обучения по дисциплине (модулю)**
10	Отлично	Студент полностью владеет знаниями, изложенными в рабочей программе, и глубоко осмысляет дисциплину. Он самостоятельно и логически последовательно отвечает на все вопросы, акцентируя внимание на наиболее важном. Умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя ключевые моменты и устанавливая причинно-следственные связи. Четко формулирует ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	
8	Отлично	
7	Хорошо	Студент обладает знаниями предмета почти в полном объеме рабочей программы и самостоятельно, логически последовательно и всесторонне отвечает на все вопросы, акцентируя внимание на наиболее значимых моментах. Он умеет анализировать, сравнивать, классифицировать, обобщать, конкретизировать и систематизировать изученный материал, выделяя его ключевые аспекты и устанавливая причинно-следственные связи. Формулирует свои ответы, уверенно интерпретирует результаты анализов и других исследований, а также решает сложные ситуационные задачи. Студент хорошо знаком с методами исследования, необходимыми для практической деятельности, и умеет связывать теоретические аспекты предмета с практическими задачами.
6	Хорошо	
5	Удовлетворительно	Студент обладает базовыми знаниями по дисциплине, но испытывает трудности при самостоятельных ответах и использует неточные формулировки. В ходе ответов он допускает ошибки, касающиеся сути вопросов. Студент способен решать только самые простые задачи и владеет лишь минимальным набором методов исследования.
4	Удовлетворительно	
3	Не сдан	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.

Дисциплина (модуль) «Функциональное программирование» оценивается следующим образом:

Активность	Вес	Количество	Описание
Домашние задания	40%	7	Набор задач по темам недели
Проект	30%	1	Исследовательская работа по дисциплине (модулю) и презентация результатов
Зачет с оценкой	30%	1	Коллоквиум – устные ответы на вопросы, список которых известен студенту

Формула расчёта итоговой оценки по дисциплине (модулю) «Функциональное программирование»: «0,4 × Среднее за домашние задания + 0,3 × Проект + 0,3 × Зачет с оценкой».

В рамках изучения дисциплины (модуля) возможно получение бонусных баллов.

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные домашние задания

Домашнее задание 1. Рекурсия, списки и доказательства по индукции

1. Реализуйте на Haskell следующие функции:
   - map, filter, foldr, foldl — без использования стандартной библиотеки
   - reverse, zipWith, takeWhile — с использованием рекурсии
   - isSorted :: Ord a => [a] -> Bool
   - merge :: Ord a => [a] -> [a] -> [a] (слияние двух отсортированных списков)
2. Реализуйте сортировку слиянием (mergeSort) и докажите по индукции, что она возвращает отсортированный список (в виде комментария в коде или отдельного текста).
3. Напишите генератор случайных списков с помощью QuickCheck и протестируйте свойства:
   - mergeSort . mergeSort = mergeSort (идемпотентность)
   - isSorted (mergeSort xs)
   - length (mergeSort xs) == length xs
4. Оформите в виде модуля с комментариями и тестами.

Домашнее задание 2. Алгебраические типы, классы типов и STM

1. Определите алгебраический тип данных BankAccount, содержащий:
   - номер счёта (String),
   - баланс (Double),
   - статус (Active, Frozen, Closed).
2. Реализуйте:
   - функции deposit, withdraw, transfer
   - безопасную версию transferSTM :: AccountId -> AccountId -> Double -> STM (), используя STM и TVar
3. Напишите тест, моделирующий 100 параллельных переводов между двумя счетами,

и покажите, что STM предотвращает состояние гонки.
4. Реализуйте Functor, Foldable, Traversable для типа Tree a и проверьте законы с помощью QuickCheck.

Домашнее задание 3. Парсеры и монады

1. Используя Applicative и Monad, напишите парсер арифметических выражений с операциями +, -, *, /, скобками и числами.
   Пример: "1 + (2 * 3)" → Expr.
2. Реализуйте интерпретатор для этого AST (абстрактного синтаксического дерева).
3. Добавьте поддержку переменных и среды (Env = Map String Double), используя ReaderT Env IO.
4. Напишите property-based тесты на QuickCheck, проверяющие, что парсер корректно разбирает и интерпретирует выражения.

Примерное описание задания и критерии оценивания к проекту

Студент разрабатывает небольшой функциональный проект на Haskell, демонстрирующий владение ключевыми концепциями: алгебраические типы, классы типов, монады, STM, парсеры, property-based testing, tagless final, GADT или другие продвинутые техники.

Примеры проектов:

- Интерпретатор простого языка программирования (например, STLC) с использованием GADT и Free-монады.
- Веб-сервис на scotty с CRUD-операциями и STM-бэкендом.
- Система обработки финансовых транзакций с транзакционной памятью и проверкой свойств.
- Парсер и анализатор логов с потоковой обработкой через streamly.
- Реализация FRP-приложения (например, таймер, калькулятор) на reactive-banana.

Этапы выполнения проекта:

1. Постановка задачи — формулировка цели и функциональных требований.
2. Проектирование — выбор архитектуры (tagless final, монадные трансформеры, STM и т.д.).
3. Реализация — написание кода с использованием продвинутых техник.
4. Тестирование — unit- и property-based тесты, проверка законов классов типов.
5. Оформление — отчёт (5–10 стр.) или README с описанием, кодом, примерами использования, выводами.

Формат: репозиторий на GitHub (или архив) + отчёт + презентация (5–7 слайдов).

Критерии оценивания проекта:

1. **Использование продвинутых концепций Haskell**
   — Применены GADT, STM, tagless final, Free-монады, линзы, refined-типы и др. в соответствии с задачей.
2. **Корректность и типобезопасность кода**
   — Программа компилируется, не содержит runtime-ошибок, использует сильную типизацию для исключения невалидных состояний.

3. **Качество архитектуры и модульности**
   — Чёткое разделение на модули, использование абстракций, композиция эффектов, соблюдение лучших практик.
4. **Наличие и качество тестирования**
   — Реализованы unit- и property-based тесты, проверены законы (например, моноид, функтор), протестированы крайние случаи.
5. **Оформление и документация**
   — Код с комментариями, README, инструкции по сборке, отчёт с пояснением решений и выводами.

Примерные вопросы к коллоквиуму

1. Что такое алгебраический тип данных? Приведите пример и объясните, как он компилируется.
2. В чём разница между Functor, Applicative и Monad? Приведите примеры.
3. Как работает программная транзакционная память (STM)? В чём её преимущество перед мьютексами?
4. Что такое tagless final и в каких случаях он предпочтительнее монадных трансформеров?
5. Как с помощью GADT можно реализовать типобезопасный интерпретатор?
6. Что такое линза Лаарховена? Как она используется для работы с вложенными структурами?
7. Как QuickCheck генерирует тестовые данные? Приведите пример проверки закона функтора.
8. Что такое refined-типы? Как они помогают избежать ошибок времени выполнения?
9. В чём суть функционального реактивного программирования (FRP)? Как работает reactive-banana?
10. Что такое Weak и StableName? Как они используются для управления памятью и предотвращения утечек?

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1.	Что такое полиморфная функция в Haskell?	Функция, работающая с разными типами через параметрический полиморфизм	УК-1
2.	Какой тип имеет выражение foldr (+) 0?	Num a => [a] -> a	УК-2
3.	Что такое моноид? Приведите пример.	Тип с ассоциативной бинарной операцией и нейтральным элементом,	ПК-1

		например, Int с + и 0	
4.	Какой класс типов позволяет применять функцию к каждому элементу структуры?	Functor	ПК-2
5.	Что возвращает выражение fmap (+1) [1][2][3]?	[2][3][4]	ПК-3
6.	Какой оператор используется в Applicative для применения функции из контекста?	<*>	ПК-5
7.	Что такое STM и для чего он используется?	Программная транзакционная память — механизм для безопасной конкурентной работы с изменяемым состоянием без блокировок	УК-1
8.	Какой тип используется для представления вычислений с эффектами ввода-вывода?	IO a	УК-2
9.	Что такое GADT и в чём его преимущество перед обычными ADT?	Обобщённые алгебраические типы данных, позволяющие точно указывать возвращаемый тип в каждом конструкторе, что обеспечивает более строгую типизацию	ПК-1
10.	Какая библиотека используется для property-based тестирования в Haskell?	QuickCheck	ПК-2
11.	Что означает deriving (Eq, Show)?	Автоматическое порождение экземпляров классов Eq и Show для типа	ПК-3
12.	Какой тип имеет выражение pure (+) <*> Just 2 <*> Just 3?	Maybe Int	ПК-5
13.	Что такое ReaderT r IO a?	Монадный стек, сочетающий возможность чтения окружения типа r и выполнения эффектов ввода-вывода	УК-1

14.	Какой эффект моделирует монада ST?	Локальное изменяемое состояние с гарантией отсутствия побочных эффектов за пределами вычисления	УК-2
15.	Что такое Free-монада?	Монада, позволяющая строить DSL, отделяя описание вычислений от их интерпретации	ПК-1
16.	Как с помощью библиотеки refined можно запретить отрицательные числа?	С использованием предиката Positive или NotNegative	ПК-2
17.	Что такое линза (lens)?	Абстракция для безопасного доступа и модификации вложенных структур данных	ПК-3
18.	Какой оператор используется для композиции функций в Haskell?	(.)	ПК-5
19.	Что такое streamly?	Библиотека для потоковой обработки данных с поддержкой ленивости, строгости и параллелизма	УК-1
20.	Как reactive-banana реализует функциональное реактивное программирование?	Через абстракции Event (события) и Behavior (поведения), позволяющие моделировать динамические системы	УК-2