ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

Решением Ученого совета
АНО ВО «Центральный университет»
«07» марта 2024 г.
Протокол № 1

Рабочая программа дисциплины (модуля)
«Функциональное программирование»

Направление подготовки: *38.03.05 Бизнес-информатика*

Направленность (профиль) подготовки: *Бизнес-аналитика*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 года*

Год набора: *2024*

ДОКУМЕНТ ПОДПИСАН
ЭЛЕКТРОННОЙ ПОДПИСЬЮ

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

Электронный документ

Москва
2024

Содержание

1. **Краткая характеристика дисциплины (модуля)** 3
2. **Перечень планируемых результатов обучения** 3
3. **Тематический план** 6
4. **Содержание дисциплины (модуля)** 7
5. **Учебно-методическое обеспечение** 8
6. **Материально-техническое обеспечение** 8
7. **Методические и оценочные материалы** 10

1. Краткая характеристика дисциплины (модуля)

Рабочая программа дисциплины (модуля) «Функциональное программирование» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по направлению подготовки 38.03.05 Бизнес-информатика, профиль Бизнес-аналитика, утвержденным приказом Министерства науки и высшего образования Российской Федерации № 838 от 29.07.2020 года.

Изучение дисциплины (модуля) »Функциональное программирование» обеспечивает более предсказуемое поведение, простоту параллелизации и возможность формального доказательства свойств программ, что делает его ключевым инструментом в разработке надёжных распределённых систем, больших данных, компиляторов и в академических исследованиях в области теории вычислений..

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 38.03.05 Бизнес-информатика, профиль Бизнес-аналитика и входит в часть, формируемую участниками образовательных отношений, Блока 1.

Дисциплина (модуль) является выборной и доступна для изучения на 3 или 4 курсе в 5, 6, 7, 8 семестрах на выбор после успешного освоения дисциплин «Основы математического анализа и линейной алгебры» или «Математический анализ» и «Линейная алгебра и геометрия» или «Математический анализ. Продвинутый уровень» и «Линейная алгебра и геометрия. Продвинутый уровень» и одну из дисциплин на выбор: «Язык программирования Python», «Язык программирования Java», «Язык программирования C++», «Язык программирования Go».

Цель изучения дисциплины (модуля): сформировать у студентов фундаментальное понимание и практические навыки применения функционального подхода к программированию, позволяющие писать чистый, модульный и легко проверяемый код.

Задачи изучения дисциплины (модуля):

— изучить λ-исчисление, типовые системы (HM, Hindley-Milner, зависимые типы) и семантику чистых функций;

— освоить основные конструкции функциональных языков (каррирование, высшие-порядковые функции, рекурсия, ленивость, монады, эффекты);

— научиться проектировать и реализовывать абстракции данных без побочных эффектов, использовать типизированные библиотеки и инструменты статической проверки;

— развить навыки отладки, тестирования и формальной верификации функциональных программ в современных IDE/REPL (Haskell, OCaml, F#, Scala).

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), best practices организации кода, 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 constraint-доказательств 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 поиск, критический анализ и синтез информации, применять системный подход для решения поставленных задач	УК-1.1.	Знает методы поиска и анализа информации в области аналитики, основные принципы критической оценки источников информации и их релевантности.
		УК-1.2.	Умеет критически оценивать источники информации и синтезировать данные из различных источников для решения задач, применять системный подход к анализу и решению комплексных проблем
		УК-1.3.	Имеет практический опыт работы с современными инструментами и технологиями для обработки информации, формулировании и структурировании задач на основе полученной информации
ОПК-4.	Способен понимать принципы работы информационных технологий; использовать информацию, методы и программные средства ее сбора, обработки и анализа для информационно-аналитической поддержки принятия управленческих решений	ОПК-4.1.	Знает основные принципы работы информационных технологий и их влияние на бизнес-процессы
		ОПК-4.2.	Умеет использовать методы и программные средства для сбора, обработки и анализа информации, обеспечивая качественную информационно-аналитическую поддержку
		ОПК-4.3.	Имеет практический опыт в применении аналитических инструментов для поддержки принятия управленческих решений в организациях
ПК-2.	Способен использовать соответствующий математический аппарат и инструментальные средства для обработки, анализа и систематизации информации по теме исследования для решения задач профессиональной деятельности	ПК-2.1.	Знает основные математические методы и инструментальные средства, применяемые для обработки и анализа информации
		ПК-2.2.	Умеет эффективно использовать математический аппарат для систематизации данных и решения профессиональных задач
		ПК-2.3.	Имеет практический опыт работы с инструментами анализа информации в рамках исследовательских проектов

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			
1	Введение в программирование на Haskell: от синтаксиса до монад	14	14	4	62	Домашнее задание
2	Продвинутое функциональное программирование	14	14	2	64	Домашнее задание Коллоквиум
	Зачет с оценкой			4		
	Итого:	***28***	***28***	***8***	***126***	
Объем дисциплины (модуля) (в ак. ч.)		***190***				
Объем дисциплины (модуля) (в зач. ед.)		***5***				

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. Многопоточность в Haskell. Software transactional memory. Веб-серверы на scotty. Livelock и stm-containers. Композиция (аппликативных) функторов. Трансформеры монад. 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 effectful. Драйверы баз данных на примере rel8. Исключения и ресурсы. Уточняющие типы на примере 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. Weak и StableName. FFI.

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 (дата обращения: 14.07.2026). – Режим доступа: по подписке.

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 (дата обращения: 14.07.2026). – Режим доступа: по подписке.

3. 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 (дата обращения: 14.07.2026). – Режим доступа: по подписке.

Дополнительная литература:

1. 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 (дата обращения: 14.07.2026). – Режим доступа: по подписке.

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 (модуле), 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:		

Электронный документ

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, коллоквиум, бонусные баллы, 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.

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 курсу 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 (модуле) «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**	**Оценка за зачет**	**Общая характеристика результата обучения по дисциплине (модулю)**
10	Отлично	Зачтено	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
9	Отлично	Зачтено	
8	Отлично	Зачтено	

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
			связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
7	Хорошо	Зачтено	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	Зачтено	
5	Удовлетворительно	Зачтено	Студент обладает базовыми знаниями по дисциплине (модулю), 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	Зачтено	
3	Не сдан	Не зачтено	Студент не овладел обязательным минимумом знаний по предмету и не может ответить на вопросы, даже если преподаватель задает дополнительные наводящие вопросы.
2	Не сдан	Не зачтено	
1	Не сдан	Не зачтено	

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

Активность	Вес	Количество	Описание
Домашние задания	40%	7	Набор задач по темам недели
Коллоквиум	30%	1	Устные ответы на вопросы, список которых известен студенту заранее
Зачет с оценкой	30%	1	Проект - исследовательская работа по курсу и презентация результатов.

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

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

Электронный документ

Текущий контроль успеваемости обучающихся по дисциплине (модуле)

Примерные домашние задания

1. **Полиморфный `map`.**

— Сформулировать типовую сигнатуру `map :: (a → b) → [a] → [b]`.

— Реализовать чистую рекурсивную функцию `map`.

— Код должен компилироваться без предупреждений.

2. **Моноид.**

— Перечислить три закона моноида.

— Реализовать тип `Sum Int` через `newtype`.

— Показать, что `mempty = Sum 0` и `(<>) = λ (Sum x) (Sum y) → Sum (x + y)`.

— Проверить законы с помощью `quickCheck`.

3. **Индукция над списками.**

— Доказать (вручную или при помощи QuickCheck) свойство `length (replicate n x) = n` для натуральных `n`.

— Предоставить базовый случай, шаг индукции и/или пройти тест для случайных значений.

4. **Алгебраический тип.**

— Объявить бинарное дерево `BinaryTree a`.

— Реализовать для него инстанцию `Functor`.

— Привести рабочий файл, компилирующийся без ошибок.

5. **Комбинаторы парсера.**

— С помощью `parsec`/ `megaparsec` написать парсер арифметических выражений (цифры, `+`, `*`).

— Функция `parseExpr :: String → Either ParseError Expr` должна правильно разбрать строки вида `2+3*4`.

6. **Applicative-композиция.**

— Показать, как два значения `Maybe` комбинируются через `<*>`.

— Привести пример `Just (+) <*> Just 3 <*> Just 4` и получить результат `Just 7`.

7. **STM-транзакция.**

— Реализовать функцию, атомарно увеличивающую счётчик `TVar Int` и одновременно записывающую старое значение в второй `TVar`.

— Протестировать работу с несколькими потоками, убедившись в отсутствии гонок.

8. **Тестирование свойства.**

— Написать QuickCheck-свойство для `reverse :: [a] → [a]`, например `reverse (reverse xs) == xs`.

— Убедиться, что тест проходит для широкого набора случайных списков.

9. **Web-сервер (Scotty).**

— Создать минимальный сервер, обслуживающий запрос `GET /hello/:name`.

— Сервер должен возвращать JSON-ответ `{ "greeting":"Hello, <name>!" }`.

10. **FFI.**

— Через FFI вызвать из Haskell функцию `double sqrt(double)` из стандартной библиотеки C.

Электронный документ

— Написать оболочку **hsSqrt :: Double → Double**, проверив, что **hsSqrt 9** возвращает **3.0**.

Примерные задания и критерии оценивания для проекта

Варианты тем проекта:

— A. DSL-интерпретатор – Описать типобезопасный DSL (арифметика, ветвление, ввод/вывод) на основе GADT и Free-монады , реализовать два интерпретатора (IO- и Mock-), покрыть код property-тестами.

Требуемые технологии : GADT, free-library, mtl (MonadIO), QuickCheck.

— B. Стрим-обработчик – Построить pipeline на streamly / conduit, читающий CSV-файл, группирующий строки, применяющий агрегирующую функцию и выводящий результат в Vector. Оценить пропускную способность.

Требуемые технологии : streamly (или conduit), vector, criterion.

— C. Сервер с STM-логикой – Реализовать веб-сервер (Scotty) с несколькими энд-поинтами, использующими STM -контейнеры для учёта пользовательских сессий, обеспечить корректность при высоком уровне параллелизма.

Требуемые технологии : scotty, stm, async, quickcheck-stm.

— D. Effect-system – Создать небольшую библиотеку эффектов на базе effectful / polysemy (или mtl), включив эффекты Reader, State, Error, Log. Продемонстрировать её на примере простой командной оболочки.

Требуемые технологии : effectful / polysemy, mtl, hspec.

— E. Тест-платформа – Сконструировать набор типовых property-тестов (QuickCheck/ Hedgehog) для реализаций Functor, Applicative, Monad; собрать отчёт о покрытии и найденных ошибках.

Требуемые технологии : quickcheck, hedgehog, tasty, hpc.

Студент выбирает один из вариантов, фиксирует задачу в Issue репозитория и получает её одобрение.

Критерии оценивания:

— Функциональность.
— Качество кода.
— Использование продвинутых концепций.
— Тестирование.
— Документация.
— Презентация.
— Отчёт.

Примерные задания для коллоквиума

1. Полиморфизм Hindley-Milner. Дайте определение, объясните разницу между универсальным полиморфизмом и ограниченным полиморфизмом, реализуемым через type-classes.

2. Моноид. Сформулируйте класс Monoid в Haskell, перечислите три закона, которые он обязан удовлетворять, и приведите пример моноидного типа.

3. Функтор и аппликативный функтор. Опишите основные операции (fmap, pure, <*>) и перечислите их законы (идент., композиция, гомоморфизм, интерпретация).

4. Do-нотация. Объясните, как она разворачивается в цепочку вызовов >>= и >>.

5. Software Transactional Memory (STM). Что представляет собой STM и как достигается атомарность транзакций?

6. GADT. В чём отличие от обычных `data`-объявлений? Приведите простой пример типобезопасного DSL, построенного на GADT.

7. Free-монда. Какие задачи решает Free-Monad, как выглядит её базовая структура в коде (конструкторы `Pure` и `Free`)?

8. Effect-system. Назовите три уровня абстракции (effect type, handler, carrier) в библиотеках `effectful` / `polysemy` и кратко опишите их функции.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1.	Запишите типовую сигнатуру и реализацию полиморфной функции **map** для списка.	`haskell\nmap :: (a -> b) -> [a] -> [b]\nmap _ [] = []\nmap f (x:xs) = f x : map f xs\n`	ПК-2
2.	Сформулируйте **законы моноида** и укажите, какой экземпляр типа `Sum` в Haskell является моноидом.	Законы: (i) `mempty <> x = x`, (ii) `x <> mempty = x`, (iii) `(x <> y) <> z = x <> (y <> z)`. Для `Sum a` (где `a` — мономо́ид) `mempty = Sum 0`, `(<>) = \(Sum x) (Sum y) -> Sum (x + y)`.	УК-1
3.	Доказать индукцией, что `length (replicate n x) = n` для любого `n :: Nat`.	Индукция по `n`: Base `n=0`: `replicate 0 x = []`, `length [] = 0`. Step: предположим для `n`, тогда `replicate (n+1) x = x : replicate n x`; `length (…) = 1 + length (replicate n x) = 1 + n = n+1`. ■	УК-1
4.	Приведите пример **Applicative-композиции** двух `Maybe`-значений.	`haskell\npure f <*> mx <*> my = case (mx,my) of\n (Just x, Just y) -> Just (f x y)\n _ -> Nothing\n-- пример: Just (+) <*> Just 3 <*> Just 4 → Just 7\n`	УК-1
5.	Реализуйте программу в **do-нотации,** которая читает строку, конвертирует её в `Int` и выводит факториал, используя `ST`-монаду для локального массива.	`haskell\nimport Control.Monad.ST\nimport Data.Array.ST\nimport System.IO\nfactorialST :: Int -> Int\nfactorialST n = runST $ do\n arr <- newArray (0,n) 1 :: ST s (STUArray s Int Int)\n mapM_ (\\i -> do v <- readArray arr (i-1)\n writeArray arr i (v*i)) [1..n]\n readArray arr n\nmain = do\n s <- getLine\n let n = read s :: Int\n print (factorialST n)\n`	ПК-2
6.	Сформулировать **свойство** для `reverse :: [a] -> [a]` и проверить его с помощью **QuickCheck** .	Свойство: `reverse (reverse xs) == xs`. QuickCheck-тест: `quickCheck $ \\xs -> reverse (reverse xs) == (xs :: [Int])`.	ПК-2
7.	Объясните, как **STM** обеспечивает атомарность транзакций; приведите пример с двумя `TVar`-ами, где их	STM — software transactional memory: операции `readTVar` / `writeTVar` выполняются в транзакции (`atomically`). Пример: `atomically $ do a <- readTVar t1; b <-`	ОПК-4

	совместное обновление должно быть необратимым.	`readTVar t2; writeTVar t1 (a+1); writeTVar t2 (b-1)`. Если вторая операция бросит исключение, обе отменятся, гарантируя атомарность.	
8.	Напишите **минимальный веб-сервер** на `scotty`, обслуживающий GET-запрос «/hello/:name» и возвращающий JSON `{"greeting":"Hello, <name>!"}`.	`haskell\n{-# LANGUAGE OverloadedStrings #-}\nimport Web.Scotty\nimport Data.Aeson (object, (.=))\nmain = scotty 3000 $\n get \"/hello/:name\" $ do\n n <- param \"name\"\n json $ object [\"greeting\" .= (\"Hello, \" ++ n ++ \"!\")]\n`	ПК-2
9.	Охарактеризуйте **livelock** в системе, использующей `STM`-контейнеры, и предложите способ его предотвращения.	Livelock — потоки постоянно перезапускают транзакцию, не завершаясь из-за конфликтов. Предотвратить можно, введя случайный back-off (`retry` + `threadDelay`) или уменьшив степень конкуренции (разбивая данные на независимые `TVar`).	ОПК-4
10.	Составьте стек **monad-transformer-ов** `ReaderT Env (StateT St IO)` и покажите, как «поднять» действие `putStrLn`.	`haskell\ntype AppM = ReaderT Env (StateT St IO)\nrunApp :: Env -> St -> AppM a -> IO (a, St)\nrunApp env st m = runStateT (runReaderT m env) st\n-- lift:\nliftIO $ putStrLn \"hello\"\n-- в AppM: lift $ lift $ putStrLn \"hello\"\n`	ПК-2
11.	Определите **GADT** для типобезопасного арифметического DSL (константы, сложение) и напишите интерпретатор.	`haskell\ndata Expr a where\n EInt :: Int -> Expr Int\n EBool :: Bool -> Expr Bool\n EAdd :: Expr Int -> Expr Int -> Expr Int\neval :: Expr a -> a\neval (EInt n) = n\neval (EBool b) = b\neval (EAdd x y) = eval x + eval y\n`	ПК-2
12.	С помощью **type families** реализуйте вычисление длины тип-уровневого списка `type family Length xs where …`.	`haskell\ntype family Length xs where\n Length '[] = 0\n Length (x ': xs) = 1 + Length xs\n`	ПК-2
13.	Используя библиотеку **refined** , объявите тип `PositiveInt` (целое > 0) и функцию, принимающую только такие значения.	`haskell\nimport Refined\nnewtype PositiveInt = PositiveInt (Refined (GreaterThan 0) Int) deriving Show\nposInc :: PositiveInt -> PositiveInt\nposInc (PositiveInt x) = PositiveInt (refine (unrefine x + 1) \\`orElse\\` error \"impossible\")\n`	ПК-2
14.	Сформулируйте **наивный линз** для типа `data Pair a b = Pair a b` и докажите два закона линзы (Get-Put, Put-Get).	`haskell\ndata Pair a b = Pair a b\n\ntype Lens s t a b = forall f. Functor f => (a -> f b) -> s -> f t\npairFst :: Lens (Pair a b) (Pair a' b) a a'\npairFst f (Pair a b) = fmap (\\a' -> Pair a' b) (f a)\n-- Get-Put: view (set v s) = v (holds by definition)\n-- Put-Get: set (view s) s = s (holds by definition)\n`	ОПК-4

15.	Продемонстрируйте **композицию Applicative-функторов** через тип `Compose`. Реализуйте `pure` и `<*>` для `Compose Maybe []`.	`haskell\nnewtype Compose f g a = Compose { getCompose :: f (g a) }\ninstance (Applicative f, Applicative g) => Applicative (Compose f g) where\n pure x = Compose (pure (pure x))\n Compose fg <*> Compose xy = Compose ((<*>) <$> fg <*> xy)\n-- пример: getCompose (pure 5 :: Compose Maybe [] Int) = Just [5]\n`	ПК-2
16.	Напишите **pipeline** на `streamly`: читаем список `[1..10]`, умножаем каждый элемент на 2, фильтруем чётные, выводим как `Vector Int`.	`haskell\nimport Streamly\nimport qualified Streamly.Prelude as S\nimport qualified Data.Vector as V\npipeline = S.fromList [1..10]\n & S.map (*2)\n & S.filter even\n & S.toList\nmain = print (V.fromList pipeline)\n`	ПК-2