ЦЕНТРАЛЬНЫЙ УНИВЕРСИТЕТ

УТВЕРЖДЕНА

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

Рабочая программа дисциплины (модуля)
«Разработка на Rust»

Направление подготовки: *02.03.01 Математика и компьютерные науки*

Направленность (профиль) подготовки: *Математика и компьютерные науки*

Квалификация (степень) выпускника: *бакалавр*

Форма обучения: *очная*

Срок освоения программы: *4 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) «Разработка на Rust» составлена в соответствии с федеральным государственным образовательным стандартом высшего образования – бакалавриат по направлению 02.03.01 «Математика и компьютерные науки», профиль «Математика и компьютерные науки», утвержденный приказом Министерства науки и высшего образования Российской Федерации № 807 от 23.08.2017 года.

Изучение дисциплины (модуля) «Разработка на Rust» формирует у обучающегося уникальные навыки разработки программного обеспечения, сочетающего производительность системных языков с гарантиями безопасности времени выполнения. Освоение Rust позволяет участвовать в создании критически важных систем, включая операционные компоненты, встраиваемые системы и высоконагруженные сервисы, где недопустимы утечки памяти и неопределённое поведение.

Место дисциплины (модуля) в структуре образовательной программы

Настоящая дисциплина (модуль) включена в учебный план по программе подготовки бакалавриата по направлению 02.03.01 «Математика и компьютерные науки», профиль «Математика и компьютерные науки» и входит в вариативную часть Блока 1, формируемую участниками образовательных отношений, как дисциплина по выбору.

Дисциплина (модуль) изучается на 3 или 4 курсе в 5,6,7 или 8 семестре по выбору. Доступна к изучению после успешного освоения одной из дисциплин (модулей): «Язык программирования Python. Базовый», «Язык программирования Python. Основной», «Язык программирования Go», «Язык программирования Java» или «Язык программирования C++».

Цель изучения дисциплины (модуля): подготовить обучающегося к разработке высоконадёжных, производительных и безопасных системных приложений с использованием современных механизмов владения памятью, многопоточности и низкоуровневого программирования на языке Rust.

Задачи изучения дисциплины (модуля):

— освоить модель памяти Rust, включая механизмы владения, заимствования и времени жизни ссылок, для предотвращения ошибок управления памятью;

— научиться применять умные указатели, внутреннюю изменяемость и системы типов для построения сложных и безопасных структур данных;

— развить навыки разработки многопоточных и асинхронных приложений с обеспечением потокобезопасности и отсутствия состояний гонки;

— освоить методы интеграции с C, написание безопасных обёрток и использование небезопасных блоков с контролем рисков;

— сформировать компетенции в проектировании и отладке системных программ, включая тестирование, профилирование и разработку под no-std окружения.

В результате освоения дисциплины (модуля), обучающийся должен:

знать:

— механизмы обеспечения безопасности памяти и предотвращения ошибок на основе владения и заимствования;

— особенности внутреннего устройства модели памяти Rust;

— классификацию и отличия механизмов полиморфизма в Rust;

— концепцию времени жизни ссылок, правила их неявного выведения и принципы управления разделяемым состоянием;

— определение безопасной многопоточности и механизмы её обеспечения для пользовательских типов;

— основные принципы модели исполнения асинхронного кода;

— методы обработки ошибок без использования исключений;

— закономерности взаимодействия Rust-кода с языком Си и границы применимости небезопасных блоков.

уметь:

— разрабатывать многофайловые проекты и настраивать модульную архитектуру с использованием рабочих пространств и системы пакетов;

— применять механизмы внутренней изменяемости и умные указатели для построения сложных графов объектов;

— использовать стандартный инструментарий экосистемы для написания модульных, интеграционных и документирующих тестов;

— оптимизировать код и размещение данных в памяти, применяя профилировщики и средства замера производительности;

— настраивать асинхронное сетевое взаимодействие и операции ввода-вывода с помощью среды выполнения асинхронного кода;

— анализировать и обрабатывать граничные случаи бизнес-логики, кодируя инварианты в систему типов;

— интегрировать существующий код на языке Си в проекты на Rust, используя средства автоматизации межъязыкового взаимодействия и создавая безопасные обёртки над небезопасными функциями;

— составлять декларативные макросы для устранения шаблонного кода.

владеть:

— навыками написания идиоматичного Rust-кода с избеганием частых антипаттернов;

— опытом проектирования надёжных системных приложений сбора и обработки данных;

— методиками предотвращения состояний гонки и взаимных блокировок с использованием каналов и примитивов синхронизации;

— практикой низкоуровневой отладки и поиска неопределённого поведения с применением профилировщиков и санитайзеров;

— способами адаптации и сборки программного обеспечения для сред без стандартной библиотеки операционной системы.

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 разработки, 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-1.	Способен консультировать и использовать фундаментальные знания в области математического анализа, комплексного и функционального анализа алгебры, аналитической геометрии, дифференциальной геометрии и топологии, дифференциальных уравнений, дискретной математики и математической логики, теории вероятностей, математической статистики и случайных процессов, численных методов, теоретической механики в профессиональной деятельности	ОПК-1.1.	Знает основные концепции и теории в области математического анализа и смежных дисциплин; методы и подходы, используемые в различных областях математики
		ОПК-1.2.	Умеет применять математические методы для решения профессиональных задач
		ОПК-1.3.	Имеет практический опыт разработки и реализация математических моделей в профессиональной деятельности
ОПК-4.	Способен находить, анализировать, реализовывать программно и	ОПК-4.1.	Знает базовые основы современного математического аппарата,

	использовать на практике математические алгоритмы, в том числе с применением современных вычислительных систем		связанного с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности
		ОПК-4.2.	Умеет использовать этот математический аппарат в профессиональной деятельности
		ОПК-4.3.	Имеет практический опыт применения современного математического аппарата, связанного с проектированием, разработкой, реализацией и оценкой качества программных продуктов и программных комплексов в различных областях человеческой деятельности
ОПК-6.	Способен разрабатывать алгоритмы и компьютерные программы, пригодные для практического применения	ОПК-6.1.	Знает алгоритмы разработки, компьютерные программы, а также алгоритмы вычислительной математики
		ОПК-6.2.	Умеет разрабатывать математические программные продукты и комплексы с использованием современных технологий программирования
		ОПК-6.3.	Имеет практический опыт разработки интеллектуальных информационных систем для визуализации результатов исследований
ПК-3.	Способен использовать методы математического и алгоритмического моделирования при решении теоретических и прикладных задач	ПК-3.1.	Знает основные методы математического и алгоритмического моделирования, а также их применение для решения теоретических и прикладных задач
		ПК-3.2.	Умеет разрабатывать и применять математические модели и алгоритмы для решения различных задач, анализируя полученные результаты
		ПК-3.3.	Имеет практический опыт использования методов математического и алгоритмического моделирования в реальных проектах или исследованиях

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 Rust, основные особенности языка	6	6		26	Домашнее задание Контрольная работа
2	Поддержка проектов на Rust	6	6		26	Домашнее задание Контрольная работа
3	Архитектура, память и многопоточность	8	8		34	Домашнее задание Контрольная работа
4	Метапрограммирование и механизмы FFI и Unsafe	10	10		42	Домашнее задание Контрольная работа
	Зачет			2		Проект
	Итого:	***30***	***30***	***2***	***128***	
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 Rust, основные особенности языка	Введение в Rust и базовые типы Владение и заимствование (Ownership & Borrowing) Структуры и перечисления
2	Поддержка проектов на Rust	Трейты и обобщения Модули и рабочее пространство Тестирование и качество
3	Архитектура, память и многопоточность	Умные указатели и внутренняя мутабельность Продвинутое управление памятью и Lifetimes Data Layout и Динамическая диспетчеризация Многопоточность с Tokio
4	Метапрограммирование и механизмы FFI и Unsafe	Углубленное изучение по прошедшему материалы Асинхронный Rust и Сети Unsafe Rust и FFI Системное программирование и Bare-Metal

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. Макнамара, Т. Rust в действии : практическое руководство / Т. Макнамара. - Санкт-Петербург : БХВ-Петербург, 2023. - 528 с. - ISBN 978-5-9775-1166-7. - Текст : электронный. - URL: https://znanium.com/catalog/product/2123400.

2. Клабник, С. Программирование на Rust / С. Клабник, К. Николс. - Санкт-Петербург : Питер, 2021. - 592 с. - ISBN 978-5-4461-1656-0. - Текст : электронный. - URL: https://znanium.ru/catalog/product/2142469.

Дополнительная литература:

1. Лонг, Т. Хороший код, плохой код : практическое руководство / Т. Лонг. - Санкт-Петербург : БХВ-Петербург, 2023. - 432 с. - ISBN 978-5-9775-1790-4. - Текст : электронный. - URL: https://znanium.com/catalog/product/2122896.

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) «Разработка на Rust» в рамках текущего контроля успеваемости используются такие виды учебной работы, как лекции, семинары, домашние задания, контрольные работы, проект, 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.

Контрольная работа – письменная работа с набором задач, которые нужно решить за ограниченное время.

Цель контрольной работы – получить специальные знания по одной или нескольким темам дисциплины и продемонстрировать навыки их практического применения.

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.

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 «Разработка на Rust»

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***, 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 связывать теоретические аспекты дисциплины (модуля) с практическими задачами.
9	Отлично	Зачтено	
8	Отлично	Зачтено	
7	Хорошо	Зачтено	Студент обладает знаниями предмета

Десятибалльная оценка	Пятибалльная оценка	Оценка за зачет	Общая характеристика результата обучения по дисциплине (модулю)
6	Хорошо	Зачтено	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.
5	Удовлетворительно	Зачтено	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	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) «Разработка на Rust» оценивается следующим образом:

Активность	Вес	Количество	Описание
Домашнее задание	20%	14	Набор задач по темам недели
Контрольная работа	40%	4	Письменная работа с набором задач, которые нужно решить за ограниченное время
Зачет	40%	1	Проект - Исследовательская работа по дисциплине (модулю) и презентация результатов

Формула расчёта итоговой оценки по дисциплине (модулю) «Разработка на Rust»: «0,2 × среднее за домашние задания + 0,4 × среднее за контрольные работы + 0,4 × зачет».

Текущий контроль успеваемости обучающихся по дисциплине (модулю)

Примерные домашние задания

Домашнее задание 1: Основы Rust, владение и структуры

1. Установите Rust и настройте среду разработки (rustup, Cargo, IDE);
2. Создайте новый бинарный проект с помощью Cargo и напишите программу, которая выводит: Привет, я изучаю Rust!;
3. Добавьте комментарий в код: // Мое первое задание на Rust;
4. Объявите структуру User с полями name (String), age (u32) и active (bool), создайте экземпляр и выведите его;
5. Продемонстрируйте концепцию владения: передайте строку в функцию и попробуйте использовать её после вызова;
6. Исправьте ошибку с помощью заимствования (&String) и поясните, почему это работает;
7. Закоммитьте изменения, отправьте в ветку feature/ownership в GitLab и создайте Merge Request.

Домашнее задание 2: Модули, трейты и тестирование

1. Создайте библиотечный проект с именем data_structures и настройте рабочее пространство (workspace);
2. Реализуйте модуль shapes, содержащий перечисление Shape и структуры Circle, Rectangle;
3. Определите трейт Area, реализуйте его для каждой фигуры;
4. Добавьте обобщённую функцию print_area<T: Area>(shape: &T), которая выводит площадь;
5. Напишите модульные тесты для проверки корректности вычисления площади;
6. Создайте документационные комментарии и сгенерируйте документацию с помощью cargo doc --open;
7. Закоммитьте в ветку feature/modules-testing и создайте Merge Request с описанием реализованных компонентов.

Домашнее задание 3: Умные указатели, многопоточность и FFI

1. Реализуйте граф объектов с помощью Rc<RefCell<T>>, чтобы продемонстрировать внутреннюю изменяемость и разделяемое владение;
2. Создайте асинхронное приложение с использованием tokio, которое каждые 3 секунды выводит сообщение в консоль;
3. Реализуйте канал mpsc, по которому один поток отправляет данные, а другой их обрабатывает;
4. Создайте небезопасный блок unsafe, в котором вызывается простая C-функция (например, strlen) через FFI;
5. Напишите безопасную обёртку над этой функцией, чтобы скрыть unsafe логику;
6. Добавьте логирование с помощью tracing или log, чтобы отслеживать выполнение задач;
7. Закоммитьте в ветку feature/ffi-concurrency и создайте Merge Request с пояснением, как обеспечена безопасность.

Примерное описание задания и критерии оценивания к проекту

Цель проекта: разработать системное приложение на языке Rust, демонстрирующее глубокое понимание модели памяти, владения, многопоточности, асинхронного выполнения и взаимодействия с низкоуровневыми компонентами, а также применение современных практик проектирования и тестирования.

Задачи, которые необходимо выполнить студенту:

— создать проект с использованием Cargo и настроить рабочее пространство (workspace) для разделения бинарного и библиотечного кода;

— реализовать структуры и перечисления для представления бизнес-модели (например, User, Order, Status);

— применить концепции владения и заимствования для безопасной передачи данных между функциями без копирования;

— реализовать трейты для определения общего поведения (например, Processable, Serializable) и использовать обобщения для универсальных функций;

— организовать код по модулям (models, services, utils) и экспортировать публичные элементы;

— использовать умные указатели (Rc, RefCell, Arc, Mutex) для управления разделяемым состоянием и внутренней изменяемостью;

— реализовать многопоточное приложение с использованием std::thread или tokio, включающее передачу данных через каналы mpsc;

— написать асинхронную задачу, которая имитирует фоновую обработку (например, отправку уведомлений или логирование);

— интегрировать unsafe-блок для вызова функции из C-библиотеки (например, strlen или getpid) и создать над ней безопасную обёртку;

— написать модульные, интеграционные и документационные тесты, запустить их с помощью cargo test;

— добавить логирование с помощью tracing или log для отслеживания выполнения ключевых операций;

— оформить код в соответствии с идиоматикой Rust, использовать clippy и rustfmt для проверки качества;

— выложить проект в GitLab, организовать ветки и создать Merge Request с описанием архитектуры и решённых задач.

Критерии оценивания проекта:

— корректное применение механизма владения, заимствования и времени жизни ссылок для обеспечения безопасности памяти;

— использование умных указателей и примитивов синхронизации для построения потокобезопасных структур данных;

— реализация многопоточного и асинхронного кода с предотвращением состояний гонки и взаимных блокировок;

— корректная интеграция с C через FFI и создание безопасных обёрток над unsafe-функциями;

— качество кода и проектной структуры: модульность, тестирование, документация, соответствие стандартам Rust.

Примерные задания для контрольной работы

1. Объясните, как модель владения (ownership) в Rust предотвращает утечки памяти и двойное освобождение. Приведите пример передачи владения при передаче строки в функцию.
2. Напишите структуру Rectangle с полями width и height типа u32. Реализуйте метод area(), который возвращает площадь. Создайте экземпляр и выведите результат.
3. Опишите разницу между &T, &mut T, String и &str. Приведите пример, где использование заимствования (&) позволяет избежать перемещения владения.
4. Реализуйте перечисление Shape с вариантами Circle(f64) и Square(f64). Напишите функцию area(&self) -> f64, вычисляющую площадь для каждого варианта.

5. Объясните, зачем нужны трейты в Rust. Реализуйте трейт Loggable, который выводит сообщение в консоль. Реализуйте его для структуры User.

6. Создайте модуль utils с функцией is_even(n: i32) -> bool. Подключите модуль в main.rs и вызовите функцию. Укажите, как настроить структуру проекта с рабочим пространством.

7. Напишите пример использования Rc<RefCell<T>> для создания разделяемого изменяемого состояния между несколькими владельцами. Объясните, зачем нужны оба типа.

8. Опишите, как lifetimes помогают компилятору проверять валидность ссылок. Приведите пример функции с аннотацией времени жизни: fn longest<'a>(s1: &'a str, s2: &'a str) -> &'a str.

9. Напишите асинхронную функцию с использованием tokio, которая каждые 2 секунды выводит в консоль "Тик". Запустите её в main с помощью tokio::spawn.

10. Объясните, зачем используется unsafe в Rust. Напишите пример вызова C-функции strlen через FFI и создайте безопасную обёртку над ней.

Задания для промежуточной аттестации по дисциплине (модулю)

№ п/п	Задание	Ответ	Компетенция
1	Объясните, как модель владения в Rust предотвращает утечки памяти и двойное освобождение. Приведите пример передачи строки в функцию.	Владение гарантирует, что каждый объект имеет одного владельца; при передаче в функцию владение перемещается.	УК-1
2	Опишите, как работает заимствование (**&T** , **&mut T**) и почему одновременно нельзя иметь изменяемую и неизменяемую ссылки.	Правила заимствования запрещают одновременный доступ для чтения и записи, чтобы избежать гонок данных.	ОПК-6
3	Как системный подход помогает при проектировании безопасной структуры данных в Rust? Перечислите этапы анализа инвариантов и выбора типов.	Анализ требований, выбор между **String** / **&str** , **Vec** / **slice** , использование трейтов и обобщений.	УК-1
4	Напишите структуру **Person** с полями **name** (String) и **age** (u32). Реализуйте метод **is_adult(&self) -> bool** .	**impl Person { fn is_adult(&self) -> bool { self.age >= 18 } }**	ПК-3
5	Объясните, зачем используются перечисления (**enum**) в Rust. Реализуйте **Result<T, E>** для обработки ошибок без исключений.	**enum Result<T, E> { Ok(T), Err(E) }** — позволяет кодировать успех и	ОПК-4

		ошибку в типе.	
6	Как логирование помогает в отладке низкоуровневых программ на Rust? Напишите пример использования **println!** или **log** для отслеживания состояния.	Позволяет отслеживать выполнение; пример: **println!("Processing user: {:?}", user);**	ОПК-6
7	Опишите, как асинхронные задачи в **tokio** повышают производительность. Приведите пример **tokio::spawn** с фоновой задачей.	**tokio::spawn(async { loop { println!("tick"); tokio::time::sleep(...) .await } });**	ПК-3
8	Какие математические и алгоритмические принципы лежат в основе распределения памяти (**data layout**) структур в Rust? Приведите пример выравнивания полей.	Порядок и размер полей влияют на размер структуры из-за выравнивания; компилятор может переупорядочивать.	ОПК-1
9	Объясните, как Git и Cargo поддерживают системный подход в разработке на Rust. Какие практики (workspaces, **Cargo.toml**, **clippy**) помогают контролировать качество?	Cargo управляет зависимостями, тестами, сборкой; **clippy** и **fmt** обеспечивают стандарты кода.	УК-1
10	Разработайте модульную архитектуру проекта на Rust с использованием рабочего пространства. Обоснуйте разделение на **core**, **utils**, **api**.	Модульность упрощает тестирование, повторное использование и сопровождение.	ОПК-6
11	Как использование **Rc<RefCell<T>>** позволяет обойти ограничения системы владения? Приведите пример графа с разделяемыми узлами.	**Rc** — подсчёт ссылок, **RefCell** — внутренняя изменяемость; используется для циклических структур.	УК-1
12	Напишите зависимость в виде трейта **Database**, реализуйте его для **MockDB** и внедрите в функцию через обобщение.	**fn process<T: Database>(db: &T) { ... }** — DI через трейты и обобщения.	ОПК-6
13	Объясните, как работает **lifetimes** в Rust. Приведите пример функции **longest<'a>(s1: &'a str, s2: &'a str) -> &'a str**.	Аннотации времени жизни помогают компилятору проверить, что ссылки остаются	ПК-3

		валидными.	
14	Как логирование помогает выявить состояние гонки в многопоточном приложении? Опишите, какие события стоит логировать.	Логировать создание потоков, передачу данных, блокировки; помогает восстановить порядок событий.	УК-1
15	Объясните, зачем используется `Arc<Mutex<T>>` в многопоточных приложениях. Приведите пример счётчика, разделяемого между потоками.	**Arc** — атомическое владение, **Mutex** — синхронизация; `let counter = Arc::new(Mutex::new(0));`	ОПК-4
16	Как системный подход помогает при проектировании отказоустойчивого FFI-интерфейса? Перечислите меры по безопасности при вызове C-кода.	Использование **unsafe** блоков, валидация входных данных, создание безопасных обёрток.	УК-1
17	Напишите функцию, которая вызывает C-функцию **strlen** через FFI и возвращает результат как **usize**. Оберните её в безопасную функцию.	`extern "C" { fn strlen(s: *const c_char) -> usize; }` + безопасная обёртка с проверкой.	ПК-3
18	Объясните, как **tokio** и асинхронные блоки (**async**/**.await**) помогают писать эффективный сетевой код. Приведите пример HTTP-запроса.	`let response = reqwest::get("https://httpbin.org/get").await?;` — не блокирует поток.	ОПК-6
19	Как можно использовать статистические данные о размерах структур для оптимизации памяти? Опишите подход к минимизации **size_of**.	Анализ **size_of**, переупорядочивание полей, использование **#[repr(C)]** или **packed**.	ОПК-1
20	Разработайте концепцию системного приложения на Rust, которое обрабатывает данные из сети, хранит их в памяти, использует каналы и логирует операции. Опишите архитектуру.	**tokio** + **Arc<Mutex<T>>** + **mpsc** + **tracing** + FFI при необходимости; bare-metal совместимость.	ОПК-6