Учебный план программы

Rust Developer. Professional

Rust Developer. Professional

Длительность курса: 150 часов

Модуль 1. Основные концепции Rust

Тема 1.
Создание проекта и структура файлов

Содержание:
- Создание нового проекта Rust: cargo new и cargo init
- Структура директорий: src/main.rs, src/lib.rs, Cargo.toml, Cargo.lock
- Назначение каждого файла и директории

Домашнее задание

Цель: научиться создавать проект и понимать, зачем нужен каждый файл.

Задание:
- Создать проект командой cargo new hello_rust
- Открыть Cargo.toml и изменить имя автора
- Запустить программу командой cargo run
- Создать проект командой cargo init в пустой папке
- Сравнить, какие файлы создались в обоих случаях

Тема 2.
Переменные, mut и константы

Содержание:
- Создание переменных с let и константы с const
- Изменяемые переменные: ключевое слово mut
- Различия между let mut и const: когда что использовать
- Shadowing: повторное объявление переменной с тем же именем

Домашнее задание

Цель: научиться работать с переменными и понимать разницу между let, let mut и const.

Задание:
- Создать переменную с let и попробовать изменить её — посмотреть ошибку компилятора
- Исправить ошибку, добавив mut
- Создать константу с const и попробовать изменить её
- Написать программу, которая считает сумму чисел от 1 до 100 с помощью изменяемой переменной

Тема 3.
Типы данных: числа, булевы, символы

Содержание:
- Скалярные типы: целые числа (i32, u64, usize), числа с плавающей точкой (f32, f64)
- Булевы значения: true и false, их применение в условиях
- Символы: тип char, отличие от строк
- Преобразование типов: as и функции parse

Домашнее задание

Цель: научиться работать с основными типами данных и преобразовывать их друг в друга.

Задание:
- Создать переменную типа f64 и вывести её значение
- Преобразовать число в строку и обратно
- Написать программу, которая проверяет, является ли число чётным
- Создать переменную типа char и вывести её код в Unicode

Тема 4.
Строки: String и &str

Содержание:
- Скалярные типы: целые числа (i32, u64, usize), числа с плавающей точкой (f32, f64)
- Булевы значения: true и false, их применение в условиях
- Символы: тип char, отличие от строк
- Преобразование типов: as и функции parse

Домашнее задание

Цель: научиться работать с основными типами данных и преобразовывать их друг в друга.

Задание:
- Создать переменную типа f64 и вывести её значение
- Преобразовать число в строку и обратно
- Написать программу, которая проверяет, является ли число чётным
- Создать переменную типа char и вывести её код в Unicode

Тема 5.
Кортежи

Содержание:
- Создание кортежей и доступ к элементам по индексу
- Деструктуризация кортежей
- Кортежи как возвращаемое значение функций

Домашнее задание

Цель: научиться создавать и использовать кортежи.

Задание:
- Создать кортеж из трёх элементов разных типов
- Вывести каждый элемент кортежа отдельно
- Написать функцию, которая возвращает кортеж из двух чисел
- Использовать деструктуризацию для получения значений из кортежа

Тема 6.
Массивы

Содержание:
- Создание массивов и доступ к элементам
- Итерация по массиву с for
- Срезы массивов: &arr[1..3]

Домашнее задание

Цель: научиться работать с массивами и срезами.

Задание:
- Создать массив из 5 чисел и вывести сумму его элементов
- Найти максимальный элемент в массиве
- Создать срез массива и передать его в функцию
- Написать программу, которая переворачивает массив

Тема 7.
Функции

Содержание:
- Объявление функций, параметры и возвращаемые значения
- Выражения и инструкции: почему нет точки с запятой в return
- Функции без возвращаемого значения

Домашнее задание

Цель: научиться создавать и использовать функции.

Задание:
- Написать функцию, которая принимает два числа и возвращает их сумму
- Написать функцию, которая проверяет, является ли число простым
- Написать функцию, которая принимает строку и выводит её в обратном порядке

Модуль 2. Модули, крейты и тесты

Тема 1.
Модули и организация кода: mod, pub, use

Содержание:
- Создание модулей с mod
- Видимость: pub и приватность по умолчанию
- Импорт с use, вложенные модули
- Разделение кода по файлам

Домашнее задание

Цель: научиться организовывать код в несколько модулей.

Задание:
- Создать модуль math с функциями add и multiply
- Вынести модуль в отдельный файл math.rs
- Создать вложенный модуль geometry внутри math
- Использовать функции из модулей в main.rs

Тема 2.
Крейты и Cargo.toml

Содержание:
- Что такое крейт: бинарный и библиотечный
- Структура Cargo.toml: [package], [dependencies]
- Версионирование зависимостей: "1.0", "^1.2", "~1.2.3"
- Команды cargo build, cargo run, cargo check

Домашнее задание

Цель: понять, как устроен крейт и как управлять зависимостями.

Задание:
- Создать библиотечный крейт командой cargo new --lib
- Добавить зависимость rand в Cargo.toml
- Использовать rand::random() для генерации случайного числа
- Запустить cargo check и cargo build

Тема 3.
Тестирование: #[test], assert!, assert_eq!

Содержание:
- Написание юнит-тестов с #[test]
- Макросы assert!, assert_eq!, assert_ne!
- Запуск тестов: cargo test
- Тестирование паник: #[should_panic]

Домашнее задание

Цель: научиться писать и запускать тесты.

Задание:
- Написать тесты для функции add из модуля math
- Написать тест, который проверяет, что функция паникует при делении на ноль
- Запустить cargo test и убедиться, что все тесты проходят
- Написать тест, который проверяет, что функция возвращает Err при некорректном вводе

Тема 4.
Документация: ///, cargo doc

Содержание:
- Документирование кода с помощью ///
- Генерация документации: cargo doc --open
- Примеры кода в документации: doctests
- Ссылки между элементами документации

Домашнее задание

Цель: научиться документировать код и генерировать документацию.

Задание:
- Добавить документацию ко всем публичным функциям модуля math
- Сгенерировать документацию командой cargo doc --open
- Добавить пример использования в документацию функции add
- Запустить doctests командой cargo test

Модуль 3. Стандартная библиотека

Тема 1.
Коллекции: Vec, HashMap, HashSet, String, &str

Содержание:
- Коллекции в стандартной библиотеке: Vec, HashMap, HashSet, String
- Создание, добавление и удаление элементов
- Итерация по коллекциям: for, iter(), into_iter()
- Выбор подходящей коллекции для задачи

Домашнее задание

Цель: научиться работать с основными коллекциями.

Задание:
- Создать Vec, добавить в него числа, отсортировать и вывести
- Создать HashMap, посчитать количество вхождений каждого слова в строке
- Создать HashSet, удалить дубликаты из Vec
- Написать функцию, которая принимает срез и возвращает уникальные элементы

Тема 2.
Итераторы: map, filter, fold, collect, for_each

Содержание:
- Итераторы: ленивые вычисления и цепочки вызовов
- Адаптеры: map, filter, take, skip, enumerate, zip
- Потребители: collect, fold, sum, count, for_each
- Написание собственного итератора

Домашнее задание

Цель: научиться использовать итераторы для обработки данных.

Задание:
- Дан Vec<i32>. Получить Vec квадратов чётных чисел
- Посчитать сумму квадратов нечётных чисел
- Найти самое длинное слово в строке
- Написать итератор, который генерирует числа Фибоначчи

Тема 3.
Option и Result: unwrap, ?, match, if let

Содержание:
- Option<T>: Some и None, когда использовать
- Result<T, E>: Ok и Err, обработка ошибок
- Оператор ?: проброс ошибок вверх по стеку
- Паттерны: match, if let, while let

Домашнее задание

Цель: научиться обрабатывать ошибки и работать с Option и Result.

Задание:
- Написать функцию, которая парсит строку в число и возвращает Result
- Использовать ? для проброса ошибки
- Написать функцию, которая принимает Option<String> и выводит значение или "пусто"
- Переписать код с unwrap() на match с обработкой ошибок

Тема 4.
Строки: String, &str, срезы, UTF-8

Содержание:
- String: владение, изменяемость, конкатенация
- &str: срезы строк, литералы
- UTF-8: почему нельзя индексировать строку по байтам
- Работа с символами: chars(), bytes(), char_indices()

Домашнее задание

Цель: научиться работать со строками и понимать кодировку UTF-8.

Задание:
- Написать функцию, которая переворачивает строку (учитывая UTF-8)
- Посчитать количество символов и байт в строке
- Написать функцию, которая извлекает подстроку по символьным индексам
- Написать функцию, которая проверяет, является ли строка палиндромом

Модуль 4. Многопоточность

Тема 1.
Потоки: thread::spawn, move-замыкания, join, Send и Sync

Содержание:
- Создание потоков с thread::spawn
- Move-замыкания: передача владения в поток
- Ожидание завершения потоков: join
- Маркерные трейты Send и Sync

Домашнее задание

Цель: научиться создавать потоки и передавать данные между ними.

Задание:
- Создать 4 потока, каждый из которых выводит свой номер
- Передать вектор в поток с помощью move
- Посчитать сумму чисел, разбив работу между 4 потоками
- Собрать результаты из всех потоков и вывести общую сумму

Тема 2.
Каналы: mpsc, передача сообщений между потоками

Содержание:
- Каналы mpsc: multi-producer, single-consumer
- Создание канала: mpsc::channel()
- Отправка и получение сообщений: send, recv, try_recv
- Итерация по получателю

Домашнее задание

Цель: научиться передавать сообщения между потоками через каналы.

Задание:
- Создать канал и отправить из потока 10 чисел
- Получить все числа в главном потоке и вывести их
- Создать 3 потока-производителя, которые отправляют свои данные в один канал
- Реализовать простой пайплайн: один поток генерирует числа, второй возводит в квадрат, третий выводит

Тема 3.
Разделяемое состояние: Arc, Mutex, RwLock

Содержание:
- Arc: атомарный счётчик ссылок, разделение данных между потоками
- Mutex: взаимное исключение, lock(), MutexGuard
- RwLock: несколько читателей или один писатель
- Deadlock: как возникает и как избежать

Домашнее задание

Цель: научиться безопасно работать с общими данными.

Задание:
- Создать счётчик, который увеличивают 10 потоков по 1000 раз
- Использовать Arc<Mutex<i32>> для безопасного доступа
- Написать программу, которая может вызвать deadlock
- Переписать её так, чтобы deadlock не возникал

Тема 4.
Атомарные типы и lock-free структуры

Содержание:
- Атомарные типы: AtomicBool, AtomicUsize, AtomicI32
- Операции load, store, fetch_add, compare_exchange
- Memory ordering: Relaxed, Acquire, Release, SeqCst
- Когда атомарные типы быстрее Mutex

Домашнее задание

Цель: научиться использовать атомарные типы для синхронизации.

Задание:
- Переписать счётчик из предыдущего задания на AtomicUsize
- Сравнить производительность Mutex и атомарного счётчика
- Реализовать флаг остановки потоков с помощью AtomicBool
- Написать простой spinlock на основе compare_exchange

Тема 5.
Async/await: Future, runtime, tokio

Содержание:
- Асинхронное программирование: зачем нужно и чем отличается от потоков
- Future: трейт, poll, Pin
- async/await: синтаксический сахар
- Runtime: tokio, запуск асинхронных задач

Домашнее задание

Цель: научиться писать асинхронный код с tokio.

Задание:
- Написать асинхронную функцию, которая ждёт N секунд и возвращает сообщение
- Запустить несколько таких функций параллельно с tokio::join!
- Написать асинхронный TCP-эхо-сервер
- Обработать несколько клиентов одновременно

Модуль 5. Архитектура

Тема 1.
Паттерн Builder

Содержание:
- Builder: пошаговое создание сложных объектов
- Builder с consuming self: цепочка вызовов
- Builder с &mut self: переиспользование
- Валидация при build(): возврат Result

Домашнее задание

Цель: научиться создавать сложные объекты через Builder.

Задание:
- Реализовать Builder для структуры HttpRequest
- Сделать цепочку вызовов (consuming self)
- Добавить валидацию: обязательные поля, корректный URL
- Написать тесты для Builder

Тема 2.
Паттерн Newtype

Содержание:
- Newtype: обёртка над существующим типом
- Зачем нужен: типобезопасность, скрытие реализации
- Реализация трейтов для newtype: Deref, From, Display
- Ограничения и когда не стоит использовать

Домашнее задание

Цель: научиться использовать newtype для типобезопасности.

Задание:
- Создать newtype Email(String) с валидацией при создании
- Создать newtype UserId(u64) и newtype OrderId(u64)
- Реализовать Display для Email
- Написать функцию, которая принимает только UserId, а не u64

Тема 3.
Паттерн State

Содержание:
- State: поведение объекта зависит от его состояния
- Реализация через enum: match по состоянию
- Реализация через трейты: динамическая диспетчеризация
- Сравнение подходов: когда что выбирать

Домашнее задание

Цель: научиться реализовывать паттерн State.

Задание:
- Реализовать State для заказа: Pending, Paid, Shipped, Delivered
- Запретить недопустимые переходы (например, Pending → Delivered)
- Реализовать State для светофора
- Написать тесты для всех переходов

Тема 4.
Паттерн Strategy

Содержание:
- Strategy: семейство алгоритмов, взаимозаменяемых
- Реализация через трейты: generic и dyn Trait
- Передача стратегии в конструктор и в метод
- Замыкания как простая стратегия

Домашнее задание

Цель: научиться реализовывать паттерн Strategy.

Задание:
- Реализовать Strategy для сортировки: BubbleSort, QuickSort
- Реализовать Strategy для валидации пароля
- Сравнить generic и dyn Trait подходы
- Написать функцию, которая принимает замыкание как стратегию

Тема 5.
Паттерн Repository

Содержание:
- Repository: абстракция над хранилищем данных
- Трейт Repository: CRUD-операции
- Реализация для in-memory хранилища (HashMap)
- Тестирование с mock-репозиторием

Домашнее задание

Цель: научиться реализовывать паттерн Repository.

Задание:
- Создать трейт UserRepository с методами find, save, delete
- Реализовать InMemoryUserRepository
- Написать сервис, который принимает репозиторий через generic
- Написать тесты с mock-репозиторием

Тема 6.
Паттерн Adapter

Содержание:
- Adapter: преобразование интерфейса одного типа в другой
- Реализация через трейты
- Adapter для внешних библиотек
- Сравнение с From/Into

Домашнее задание

Цель: научиться реализовывать паттерн Adapter.

Задание:
- Создать трейт Logger и адаптер для стороннего логгера
- Создать адаптер для работы с файлом как с Logger
- Написать функцию, которая принимает любой Logger
- Написать тесты с mock-логгером

Тема 7.
Паттерн Facade

Содержание:
- Facade: простой интерфейс к сложной подсистеме
- Скрытие деталей реализации
- Facade как точка входа в библиотеку
- Баланс между простотой и гибкостью

Домашнее задание

Цель: научиться создавать фасады для сложных подсистем.

Задание:
- Создать фасад для работы с базой данных (connection, query, transaction)
- Скрыть детали пула соединений за простым API
- Написать пример использования фасада
- Написать тесты для фасада

Тема 8.
Паттерн Command

Содержание:
- Command: инкапсуляция запроса как объекта
- Реализация через трейты
- Undo/redo: хранение истории команд
- Макрокоманды: составные команды

Домашнее задание

Цель: научиться реализовывать паттерн Command.

Задание:
- Создать трейт Command с методами execute и undo
- Реализовать команды для текстового редактора (insert, delete)
- Реализовать undo/redo с историей команд
- Написать тесты для всех команд

Модуль 6. Фреймворки и Библиотеки

Тема 1.
Фреймворк Actix Web

Содержание:
- Архитектура и производительность Actix Web
- Маршрутизация, обработчики, извлекатели (extractors)
- Middleware: логирование, аутентификация, CORS
- Состояние приложения (Application State)

Домашнее задание

Цель: освоить создание веб-приложений с Actix Web.

Задание:
- Создать REST API для управления задачами (CRUD)
- Использовать базу данных (например, SQLite через sqlx)
- Добавить middleware для логирования запросов
- Написать интеграционные тесты для API

Тема 2.
Фреймворк Rocket

Содержание:
- Философия Rocket: декларативный подход
- Маршруты, параметры пути и запроса
- Формы и JSON: десериализация данных
- Управляемое состояние (Managed State)

Домашнее задание

Цель: освоить создание веб-приложений с Rocket.

Задание:
- Создать REST API для управления задачами (CRUD)
- Использовать базу данных (например, SQLite через sqlx)
- Добавить middleware для логирования запросов
- Написать интеграционные тесты для API

Тема 3.
Фреймворк Axum

Содержание:
- Архитектура Axum: Tower и экосистема middleware
- Маршрутизация и извлекатели (extractors)
- Обработка ошибок и кастомные ответы
- Состояние приложения и внедрение зависимостей

Домашнее задание

Цель: освоить создание веб-приложений с Axum.

Задание:
- Создать REST API для управления задачами (CRUD)
- Использовать базу данных (например, SQLite через sqlx)
- Добавить middleware для логирования запросов
- Написать интеграционные тесты для API

Тема 4.
Работа с базами данных

Содержание:
- Асинхронные драйверы баз данных: sqlx, tokio-postgres
- Пулы соединений и их настройка
- Миграции схемы базы данных
- Безопасность: защита от SQL-инъекций

Домашнее задание

Цель: научиться работать с базами данных в веб-приложениях.

Задание:
- Настроить подключение к PostgreSQL через пул соединений
- Создать миграции для таблиц users и posts
- Реализовать CRUD-операции для пользователей
- Написать тесты с использованием тестовой базы данных

Тема 5.
Аутентификация и авторизация

Содержание:
- Аутентификация: JWT, сессии, OAuth2
- Хеширование паролей: argon2, bcrypt
- Авторизация: роли, права доступа
- Защита эндпоинтов: middleware и extractors

Домашнее задание

Цель: реализовать безопасную аутентификацию и авторизацию.

Задание:
- Реализовать регистрацию и вход пользователей
- Хешировать пароли с помощью argon2
- Выдавать и проверять JWT-токены
- Защитить эндпоинты: только авторизованные пользователи могут создавать задачи

Тема 6.
Документирование API: OpenAPI, Swagger

Содержание:
- Спецификация OpenAPI: структура и основные элементы
- Генерация документации из кода: utoipa, aide
- Swagger UI: интерактивная документация
- Версионирование API

Домашнее задание

Цель: научиться документировать API и генерировать спецификацию.

Задание:
- Добавить аннотации ко всем эндпоинтам
- Сгенерировать спецификацию OpenAPI
- Подключить Swagger UI по адресу /docs
- Описать схемы запросов и ответов

Тема 7.
Развертывание: Docker, CI/CD, мониторинг

Содержание:
- Docker: multi-stage build для минимального образа
- Конфигурация через переменные окружения
- CI/CD: GitHub Actions для тестов и деплоя
- Мониторинг: метрики Prometheus, логирование

Домашнее задание

Цель: научиться упаковывать и развертывать приложение.

Задание:
- Написать Dockerfile с multi-stage build
- Настроить docker-compose с приложением и базой данных
- Создать GitHub Actions workflow для тестов и сборки образа
- Добавить эндпоинт /metrics с базовыми метриками

Модуль 7. Проектная работа

Тема 1.
Выбор темы, архитектура и планирование проекта

Содержание:
- Выбор темы проекта: требования к сложности и оригинальности
- Декомпозиция задачи: разбиение на модули и компоненты
- Выбор крейтов: оценка зависимостей
- Архитектурная диаграмма: как взаимодействуют части системы

Домашнее задание

Цель: выбрать тему, спланировать архитектуру и составить план работы.

Задание:
- Выбрать тему проекта и описать его назначение в 3–5 предложениях
- Составить список основных модулей и их ответственности
- Выбрать ключевые крейты и обосновать выбор
- Нарисовать архитектурную диаграмму (от руки или в любом инструменте)

Тема 2.
Защита проекта и финальный код-ревью

Содержание:
- Подготовка презентации: структура, ключевые слова, демонстрация
- Защита проекта: ответы на вопросы преподавателя и одногруппников
- Код-ревью: взаимная проверка работ студентов
- Рефлексия: что получилось, что можно улучшить

Домашнее задание

Цель: подготовить и защитить финальный проект, провести код-ревью работ одногруппников.

Задание:
- Подготовить презентацию проекта (5–7 слайдов)
- Защитить проект перед преподавателем и группой
- Провести код-ревью работы одного из одногруппников
- Написать рефлексию: что получилось, что можно улучшить