

Rust Developer – 100+ Lab Exercises Roadmap

□ Basic Level (30+ Exercises)

Objective: Build foundational skills in Rust syntax, ownership model, and simple applications.

Rust Fundamentals

- Set up Rust development environment (Rustup, Cargo).
- Understand Rust syntax: variables, data types, functions.
- Practice control flow: if, match, loops.
- Implement basic ownership and borrowing concepts.
- Work with references and lifetimes basics.

Data Structures & Collections

- Use Rust primitive data structures (arrays, tuples).
- Explore vectors, strings, and slices.
- Practice using hash maps and sets.
- Implement enums and pattern matching.

Error Handling

- Learn Result and Option types.
- Implement basic error handling with `unwrap`, `expect`.
- Create custom error types with `enum`.

Modules & Crates

- Organize code using modules and packages.
- Create and publish a simple crate to crates.io.

□ Intermediate Level (40+ Exercises)



TRAINING TRAINS

IT Services and IT Consultancy \ Internship-Inplant Training \ Software Training Institute

96985 48633
90250 10144

உங்கள் கடை மற்றும் வியாபாரத்திற்கு ஏற்ற **Website & Andriod App** மற்றும் **Software** மிகச்சிறந்த முறையில் செய்து கொடுக்கப்படும்.

Expertise : **AI, Machine Learning & Data Science \ Cybersecurity & Ethical Hacking**
IoT, Blockchain & Cloud Computing \ Full Stack Web & Mobile Development
Digital Marketing & SEO

W3 App Developers
332, MULLAMPARAPPU, N.G.PALAYAM POST
ERODE - 638 115
DomainHostly.COM

Objective: Master advanced Rust concepts, concurrency, and ecosystem tools.

Advanced Ownership & Lifetimes

- Deep dive into borrowing rules and lifetime annotations.
- Practice lifetime elision and complex scenarios.

Traits & Generics

- Define and implement custom traits.
- Use generics in functions, structs, and enums.
- Understand trait bounds and where clauses.

Concurrency & Async Programming

- Implement multi-threading with `std::thread`.
- Use channels for message passing.
- Build asynchronous applications using `async/await`.
- Work with `tokio` or `async-std` runtimes.

Testing & Documentation

- Write unit tests and integration tests.
- Use `Rustdoc` to generate documentation.
- Practice Test-Driven Development (TDD).

Ecosystem & Tooling

- Use Cargo features: workspace, profiles, dependencies.
- Manage versions and semantic versioning.
- Use Clippy and Rustfmt for linting and formatting.

□ Advanced Level (40+ Exercises)

Objective: Achieve expertise in systems programming, unsafe Rust, and ecosystem integration.

Unsafe Rust & FFI

- Understand when and how to use `unsafe`.



TRAINING TRAINS

IT Services and IT Consultancy \ Internship-Inplant Training \ Software Training Institute

96985 48633
90250 10144

உங்கள் கடை மற்றும் வியாபாரத்திற்கு ஏற்ற **Website & Andriod App** மற்றும் **Software** மிகச்சிறந்த முறையில் செய்து கொடுக்கப்படும்.

Expertise : **AI, Machine Learning & Data Science \ Cybersecurity & Ethical Hacking**
IoT, Blockchain & Cloud Computing \ Full Stack Web & Mobile Development
Digital Marketing & SEO

W3 App Developers

332, MULLAMPARAPPU, N.G.PALAYAM POST
ERODE - 638 115

DomainHostly.COM

- Work with raw pointers and manual memory management.
- Interface with C libraries via FFI.

Macros & Metaprogramming

- Write declarative (macro_rules!) macros.
- Create procedural macros for code generation.

Performance Optimization

- Profile Rust applications using tools like `perf`, `valgrind`.
- Optimize code for speed and memory usage.

Embedded & Systems Programming

- Develop firmware-level Rust applications (e.g., on Raspberry Pi).
- Use `no_std` environment for bare-metal programming.

WebAssembly & Networking

- Compile Rust to WebAssembly and integrate with JavaScript.
- Build network applications using libraries like `hyper` and `request`.

Large-Scale Projects & Ecosystem Contribution

- Contribute to open source Rust projects.
- Architect and design large Rust codebases.
- Implement CI/CD pipelines for Rust projects.

Recommended Tools & Libraries

- Rustup, Cargo (package manager)
- Clippy (linting), Rustfmt (formatting)
- Tokio, async-std (async runtimes)
- Serde (serialization/deserialization)
- Actix, Rocket (web frameworks)
- Criterion.rs (benchmarking)