

Q1. What are the core design goals and primary use cases for Rust?

Answer:

Q2. Explain the type system in Rust: static vs dynamic typing, type inference, and type safety.

Answer:

Q3. How does Rust handle memory management: manual, garbage collection, or reference counting?

Answer:

Q4. Describe the concurrency model in Rust: threads, async/await, coroutines, or actors.

Answer:

Q5. What are the key data structures in Rust's standard library and when do you use each?

Answer:

Q6. How does Rust implement object-oriented programming, functional programming, or both?

Answer:

Q7. Explain error handling in Rust: exceptions, Result/Either types, or error codes.

Answer:

Q8. How does Rust's package/module system work and what is the standard build tool?

Answer:

Q9. What are common performance optimization techniques specific to Rust?

Answer:

Q10. How do you write unit tests and measure code coverage in a Rust project?

Answer: