# Rust Programming Masterclass

## Course Summary Table

| | |
|---|---|
| **Duration:** | 4 Days |
| **Target Audience:** | C/C++ developers, C#/Java developers |
| **Objectives:** | • Understand the Rust language fundamentals<br>• Write idiomatic Rust code<br>• Use Rust types effectively<br>• Understand Rust ownership and lifetimes<br>• Use modules to organize code |
| **Pre Requisites:** | • Good knowledge and experience with at least one programming language such as C++, C#, Java, Python, JavaScript, Kotlin, Visual Basic, F#, … |

Instructor: **Pavel Yosifovich**

## Abstract

The Rust programming language promises to be safe, fast and productive. Created by Mozilla, Rust provides high level features while maintaining control and safety for low level code if required. Rust plays in the same playing field as C/C++ but is fit for any kind of software, from low-level system code to servers, clients and anything in between.
This course introduces the fundamentals of Rust, from the basics up to using types, modules, generics, pattern matching, error handling and more. Lab exercises are used to help sink in the theoretical concepts.

## Syllabus

- Module 1: Introduction to Rust
  - Why Rust?
  - Hello, Rust!
  - Tools
  - The Rust compiler
  - Cargo
  - Summary

- Module 2: Variables and Data Types
  - Variables
  - Mutability
  - Fundamental Data Types
  - Tuples
  - Arrays
  - Functions
  - Control Flow
  - Summary

- Module 3: Ownership
    - Why ownership?
    - References
    - Borrowing
    - Slices
    - Summary

- Module 4: Compound Types
    - Structs
    - Creating Objects
    - Traits
    - Implementing traits
    - Enums
    - Pattern Matching
    - Methods
    - Summary

- Module 5: Common Types and Collections
    - Strings
    - Vectors
    - Hash maps
    - Summary

- Module 6: Managing Projects
    - Packages and Crates
    - Using crates.io
    - Crates and Modules
    - Visibility and Scope
    - Modules and Files
    - Tests
    - Summary

- Module 7: Error Handling
    - Types of Errors
    - Panicking
    - Using the Result type
    - Unwrap and Expect
    - Propagating Errors
    - When to panic
    - Summary

- Module 8: Generics and Traits
    - Generic Data Types
    - Traits
    - Implementing traits
    - Common traits in the standard library
    - Polymorphism with traits
    - Summary

- Module 9: Smart Pointers
    - The need for smart pointers
    - The Box type
    - RC and ARC
    - Interior Mutability
    - Summary

- Module 10: Functional Programming
    - The Iterator trait
    - Closures
    - The Function Types
    - Summary

- Module 11: Threads and Concurrency
    - Processes and Threads
    - Creating Threads
    - Threads and Ownership
    - Thread Communication
    - Message Passing
    - Thread Synchronization
    - Send and Sync Traits
    - Summary

- Module 12: Async and Await
    - Asynchronous vs. Synchronous
    - Async Functions
    - Awaiting
    - Using the Tokyo Crate
    - Summary

- Module 13: Unsafe Rust and Interoperability
    - Safe vs. Unsafe Rust
    - Unsafe Usage
    - Interoperability with C
    - Interoperability with C++
    - Summary

- Module 14: Advanced Topics
    - Lifetimes
    - Lifetime Annotations
    - Lifetime Elision
    - Macros
    - Declarative Macros
    - Procedural Macros
    - Summary