

## Rust Programming

### Audience Course Rust Programming

The course Rust Programming is intended for developers who want to learn how to program in Rust and others who want to understand Rust code.

### Prerequisites Course Rust Programming

Experience programming in a modern programming language is desirable and beneficial to a good understanding.

### Realization Training Rust Programming

The theory is treated on the basis of presentation slides. The concepts are illustrated with demos. The theory is interspersed with exercises. The course times are from 9.30 to 16.30.

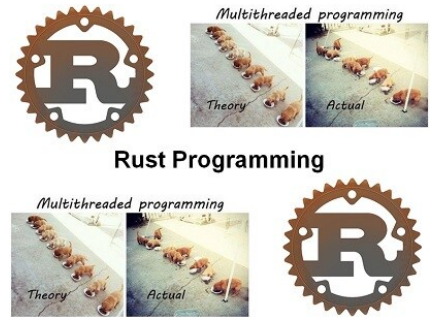
### Certification Rust Programming

The participants receive an official certificate Rust Programming after successful completion of the course.

Duration: 3 days

Price: € 1999

[Open Schedule](#)



## Content Rust Programming

In the course Rust Programming participants learn to develop software with the latest version of the innovative programming language Rust.

### Rust Fundamentals

Rust is a new, practical system programming language that produces lightning fast code. Rust is community driven. With Rust you prevent almost all crashes and data races.

### Imperative and Functional

Rust builds on a rich history of programming languages. It is low-level language with multiple paradigms, both imperative and functional.

### Concurrency and High Performance

Rust focuses on safe, high-performance, concurrent applications. Rust began to gain momentum in the industry before the official 1.0 version in May 2015, because there is a clear need for a new low-level system language.

### Traits, Borrowing en Lifetimes

This course deals with what makes Rust so unique and applies this to practical problems of system programming. Topics that will be discussed are: traits, generics, memory safety, move semantics, borrowing and lifetimes.

### Closures en Concurrency

And also the rich macro-system of Rust, closures and concurrency are discussed.

## Modules Rust Programming

Module 1 : Rust Intro	Module 2 : Data Types	Module 3 : Flow Control
What is Rust? Rust Background Rust Momentum Rust Usage Comparisons to C Rust Applications Hello Rust Comments Formatted Printing Debug and Display Literals Operators	Primitives Tuples and Arrays Slices Custom Types Enums Constants Variable Bindings Scope Shadowing Casting Inference Alias	Expressions Flow Control if else loop Nesting and labels while for and range match Guards Binding if let while let
Module 4 : Functions	Module 5 : Modules	Module 6 : Generics
Methods Closures Capturing As Input Parameters Input Functions Type Anonymity As Output Parameters Examples from std Iterator::any Iterator::find Higher order Functions	Visibility Struct Visibility use Declaration Using super Using self File Hierarchy Crates Attributes Extern crate Dead Code Custom	Functions Implementations Parametrization over Types Traits Bounds Multiple Bounds Where Clauses Associated Items Associated Types Phantom Type Parameters Unit Clarification
Module 7 : Scoping	Module 8 : Traits	Module 9 : Standard Library
RAI Ownership and Moves Functions and Methods Mutability Borrowing and Freezing Aliasing ref Pattern Lifetimes Explicit Annotation Bounds and Coercion Static Elision	Zero cost Abstraction Traits are interfaces Derive Operator Overloading Drop Iterators Clone Designators Overload and Repeat Unsafe Operations Static dispatch Dynamic dispatch	Box, stack, heap Data Structures Vectors Strings Hashmap Threads Channels Path File I/O Pipes Wait Arguments Meta