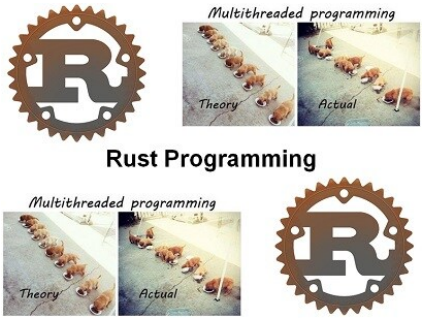
Rust Programmeren

Duur: 3 dagen **Prijs: € 1999**

Open Rooster



Doelgroep Cursus Rust Programmeren

De cursus Rust Programmeren is bedoeld voor developers die in Rust willen leren programmeren en anderen die Rust code willen begrijpen.

Voorkennis Rust Programmeren

Ervaring met programmeren in moderne programmeertaal is wenselijk en is bevorderlijk voor een goede begripsvorming.

Uitvoering Training Rust Programmeren

De theorie wordt behandeld aan de hand van presentatie slides. De concepten worden toegelicht met demo's. De theorie wordt afgewisseld met oefeningen. De cursustijden zijn van 9.30 tot 16.30.

Certificering Rust Programmeren

De deelnemers krijgen na het goed doorlopen van de cursus een officieel certificaat Rust Programmeren.

Inhoud Cursus Rust Programmeren

In de cursus Rust Programmeren leren de deelnemers software te ontwikkelen met de laatste versie van de innovatieve programmeer taal Rust.

Rust Fundamentals

Rust is een nieuwe, praktische systeem programmeertaal die razendsnelle code oplevert. Rust is community driven. Met Rust voorkom je vrijwel alle crashes en data races.

Imperatief en Functioneel

Rust borduurt voort op een rijke geschiedenis van programmeer talen. Het is low-level taal met meerdere paradigma's, zowel imperatief als functioneel.

Concurrency en High Performance

Rust richt zich op veilige, high-performance, concurrent applicaties. Rust begon al voor de officiële 1.0 versie mei 2015 momentum in de industrie te krijgen, want er bestaat een duidelijke behoefte aan een nieuwe low-level systeem taal.

Traits, Borrowing en Lifetimes

In deze cursus wordt behandeld wat Rust zo uniek maakt en wordt dit toegepast op praktische problemen van systeem programmering. Onderwerpen die aan de orde zullen komen zijn : traits, generics, memory safety, move semantics, borrowing and lifetimes.

Closures en Concurrency

En ook het rijke macro-systeem van Rust, closures en concurrency komen aan de orde.

Modules Cursus Rust Programmeren

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

SpiralTrain BV
Standerdmolen 10, 2e verdieping
3995 AA Houten

info@spiraltrain.nl
www.spiraltrain.nl
Tel.: +31 (0) 30 – 737 0661

Locaties
Houten, Amsterdam, Rotterdam, Eindhoven, Zwolle, Online