C++ 11-14-17-20

Duration: 2 days **Price: € 1499**

Open Schedule

C++ 11-14-17-20

Audience Course C++ 11-14-17-20

The course C++ 11-14-17-20 is intended for **C++** developers who want to learn the new features of the C++ 11, 14, 17 and 20 standards.

Prerequisites Course C++ 11-14-17-20

To participate in this course knowledge of and experience with **programming in C++** is required.

Realization Training C++ 11-14-17-20

The theory is discussed on the basis of presentation slides and is interchanged with exercises. Illustrative demos are used to clarify the concepts discussed. Course times are from 9.30 to 16.30.

Certification course C++ 11-14-17-20

After successful completion of the training, participants receive an official certificate C++ 11-14-17-20.

Content Course C++ 11-14-17-20

In the course **C++11**-14-17-20 the participants learn the new features of the C++ standard that have been added since C++11. The language C++ is indispensable in the world of high performance and embedded software.

C++ Standard

Since the C++98 standard it has remained silent for a long time, but since C++11 C++ has undergone a steady development in recent years with the releases of C++14, C++17 and C++20.

C++11 Features

The course starts with an overview of a number of new features in C++11 such as type inference, uniform initialization, lambda functions and strongly typed enums.

Rvalue Reference

Next the new Rvalue reference types in C++11 are discussed, which enable move constructors instead of copy constructors. The Golden Rule of 5 for classes with pointer members is also discussed.

Smart Pointers

An important addition to the C++ standard introduced with C++11 are the smart pointers. These are important in preventing memory leaks and unique, shared and weak smart pointers are discussed.

Multiple Threads

Furthermore attention is paid to multiple threads that have been part of the C++ standard since C++11. With threads C++ applications can be parallelized and make optimal use of multiple cores. The synchronization of threads by locking mechanisms such as mutexes and condition variables are also part of the course program.

C++14 Features

The new features of C++14 including generalized lambda captures, return type deduction, shared mutexes and shared locking are then discussed.

C++17 and C++20 Features

Finally attention is paid to changes and additions to the standard that have come with the release of C++17 and C++20 such as folding expressions, optional types and immediate functions.

Modules Course C++ 11-14-17-20

Module 1 : C++ 11 Intro	Module 2 : Move Semantics	Module 3 : Smart Pointers
C++11 Features	Lvalues and Rvalues in C++	unique_ptr
Type Inference	Reference to Constant	Using unique_ptr
Auto and decltype	Passing References	Specialization for Arrays
Uniform Initialization	References as Return Values	Replacement for std::auto_ptr
Initializer Lists	Rvalue References	std::make_unique
Range Based for Loop	Rvalue Reference Usage	shared _ptr Pointer
Null Pointer Constant	Assignment Operator	Control Block
Standard Types	Copy Constructor	shared_ptr Destruction Policy
constexpr Keyword	Passing and Returning Objects	shared_ptr Interface
Static Asserts	Passing References to Objects	Cyclic References
Lambda Functions	Move Constructor	std::enable_shared_from_this
Strongly Types Enums	Move Semantics	weak_ptr
User Defined Literals	Move Assignment Operator	default Keyword
Raw String Literals	Golden Rule of 5	delete Keyword
Module 4 : Multiple Threads	**Module 5 : Synchronization**	**Module 6 : C++ 14**
Multiple Threads	Data Corruption	Binary Literals
Benefits and Drawbacks	Lock Guard	Generic Lambda Expressions
Thread Characteristics	Automatic Lock Management	Generalized Lambda Captures
Thread Class	Mutex and RAII	Lambda Capture Initializers
Simple Threads	Recursive Locking	Return type Deduction
Joining Threads	Timed Locking	decltype(auto)
Detaching Threads	Atomic Types	Constraints constexpr Functions
Thread ID	Call Once	Variable Templates
Callables	Event Handling	[[deprecated]] Attribute
Passing Parameters	Condition Variables	Digit Separators
Pass by Reference	Wait and Notify	Shared Mutexes
Pass by std::ref and std::move	Promises and Futures	Shared Locking
Member Function as Thread	Asynchronous Tasks	Tuple Addressing via Type
Thread Local Storage	Working with async	std::is_final
Module 7 : C++ 17	**Module 8 : C++ 20**	
Folding Expressions	Concepts Library	
constexpr Lambda	Designated Initializers	
const if Expressions	likely and unlikely Attributes	
Init-statement for if	Immediate Functions	
Inline Variables	constexpr Virtual Functions	
Nested Namespaces	Explicit Bool	
Structured Bindings	Template Syntax Lambdas	
Selection Initialization	Math Constants	
UTF-8 Character Literals	Synchronized Outputstreams	
Initialization of Enums	std::is_constant_evaluated	
fallthrough and nodiscard	std::span	
Guaranteed Copy Elision	std::bit_cast	
Optional Types	std::midpoint	

SpiralTrain BV
Standerdmolen 10, 2e verdieping
3995 AA Houten

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

Locations
Houten, Amsterdam, Rotterdam, Eindhoven, Zwolle, Online