

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.

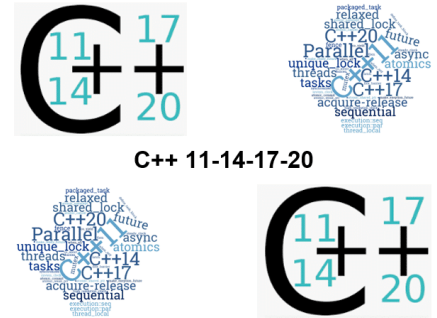
Certification course C++ 11-14-17-20

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

Duration: 2 days

Price: € 1499

[Open Schedule](#)



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