

C++ 11-14-17-20

Doelgroep Cursus C++ 11-14-17-20

De cursus C++ 11-14-17-20 is bedoeld voor **C++** developers die de nieuwe features van de C++ 11, 14, 17 en 20 standaarden willen leren kennen.

Voorkennis Cursus C++ 11-14-17-20

Om aan deze cursus te kunnen deelnemen is kennis van en ervaring met **programmeren in C++** vereist.

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

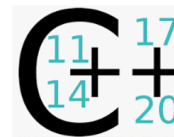
De theorie wordt behandeld aan de hand van presentatie slides en wordt afgewisseld met oefeningen. Illustratieve demo's worden gebruikt om de behandelde begrippen te verduidelijken. De cursus tijden zijn van 9.30 tot 16.30.

Certificering cursus C++ 11-14-17-20

De deelnemers krijgen na het goed doorlopen van de training een officieel certificaat C++ 11-14-17-20.

Duur: 2 dagen

Prijs: € 1499

[Open Rooster](#)


C++ 11-14-17-20



Inhoud Cursus C++ 11-14-17-20

In de cursus **C++11-14-17-20** leren de deelnemers de nieuwe features die aan de C++ standaard zijn toegevoegd sinds C++11. De taal C++ is niet weg te denken uit de wereld van high performance en embedded software.

C++ Standaard

Sinds de C++ 98 standaard bleef het lange tijd stil, maar sinds C++11 maakt C++ de laatste jaren een gestage ontwikkeling door met de releases van C++14, C++17 en C++20.

C++11 Features

De cursus gaat van start met een overzicht van een aantal nieuwe features in C++11 zoals type inference, uniform initialization, lambda functions en strongly typed enums.

Rvalue Reference

Vervolgens wordt ingegaan op de nieuwe Rvalue reference types in C++11 waardoor move constructors in plaats van copy constructors mogelijk worden. Ook wordt daarbij de Golden Rule of 5 voor classes met pointer members besproken.

Smart Pointers

Een belangrijke toevoeging aan de C++ standaard die met C++11 is geïntroduceerd zijn de smart pointers. Deze zijn van belang bij het voorkomen van memory leaks en zowel unique, shared en weak smart pointers komen aan de orde.

Multiple Threads

Verder is er aandacht voor multiple threads die sinds C++11 onderdeel zijn van de C++ standaard. Met threads kunnen C++ applicaties geparalleliseerd worden en optimaal gebruik maken van multiple cores. Ook de synchronisatie van threads door locking mechanismen als mutexes en condition variables zijn onderdeel van het cursus programma.

C++14 Features

De nieuwe features van C++14 waaronder generalized lambda captures, return type deduction, shared mutexes en shared locking komen vervolgens aan bod.

C++17 en C++20 Features

Tenslotte is er aandacht voor wijzigingen en toevoegingen aan de standaard die gekomen zijn met de release van C++17 en C++20 zoals folding expressions, optional types en immediate functions.

Modules Cursus 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	