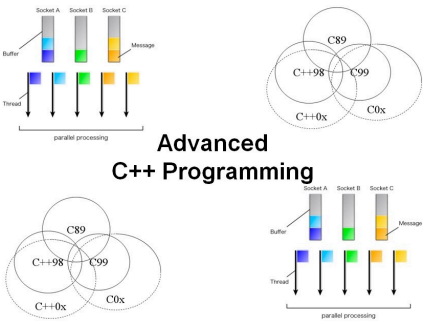
Advanced C++ Programmeren

Duur: 4 dagen **Prijs: € 2850**

Open Rooster



Doelgroep Cursus Advanced C++ Programmeren

De cursus Advanced C++ is bedoeld voor developers die bekend zijn met C++ maar die zich willen verdiepen in de meer geavanceerde technieken en mogelijkheden van de taal.

Voorkennis Cursus Advanced C++ Programmeren

Om aan deze cursus deel te nemen is kennis van de basis concepten van C++ en uitgebreide ervaring met programmeren in C++ vereist. De deelnemers moeten vertrouwd zijn met onderwerpen als operator overloading, templates, virtual functions en polymorfisme.

Uitvoering Training Advanced C++

De theorie wordt behandeld aan de hand van presentatie slides. Illustratieve demo's worden
gebruikt om de behandelde concepten te verhelderen. Er is een sterke afwisseling van theorie en praktijk en ruime gelegenheid om te oefenen.

Officieel Certificaat Advanced C++ Programmeren

De deelnemers krijgen na het goed doorlopen van de cursus een officieel certificaat Advanced C++ Programmeren.

Inhoud Cursus Advanced C++ Programmeren

In de cursus Advanced C++ komen de nieuwe en geavanceerde aspecten van de C++ taal gebaseerd op de standaarden C++11, C++14, C++17 en C++20 ruimschoots aan de orde.

C++11 Features

De cursus gaat van start met een overzicht van de features die in C++11 zijn geïntroduceerd zoals type inference, initializer lists, range based for loop, lambda functies en strongly typed enums.

Right References en Move Constructors

Vervolgens wordt ingegaan op right references en de performance winsten die geboekt kunnen worden door move constructors naast copy constructors te gebruiken.

Inheritance Toevoegingen

Ook de nieuwe mogelijkheden met betrekking tot inheritance met de keywords default, delete, override en final komen aanbod. En passant wordt ingegaan op de implementatie van virtual functions en de noodzaak van virtual destructors.

Smart Pointers

Daarnaast wordt in detail gekeken naar smart pointers en hierbij komen unique pointers, shared pointers en weak pointers aan de orde.

Operator Overloading en Templates

Ook operator overloading en templates staan op het programma, waarbij ingegaan wordt op variadic templates en perfect forwarding.

RAII Pattern

Het modern C++ Resource Acquisition is Initialization ofwel RAII idioom komt aan de orde bij exception handling.

Multithreading

Threads evenals de synchronisatie tussen threads zijn onderdeel van de standaard en worden besproken. Hierbij wordt ook ingegaan op asynchrone calls met promises en futures.

C++11, C++17 en C++20 Features

Vervolgens komen specifieke C++11, C++17 en C++20 features aan bod zoals optional types, structured binding declarations en constructies uit de wereld van functioneel programmeren zoals fold expressions.

Advanced STL

De cursus besluit met een overzicht van geavanceerde mogelijkheden van de Standard Template Library **STL**.

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

Modules Cursus Advanced C++ Programmeren

Module 1 : Modern C++ Features	Module 2 : Move Semantics	Module 3 : Inheritance
C++11 Features	Reference Initialization	default and delete Keyword
Type Inference	References and Pointers	Delegating Constructors
Auto Keyword	Rvalues and Rvalues in C++	Inheritance
Deduction with decltype	Passing and Returning References	Calling Base Class Constructors
Uniform Initialization	Rvalue References	Multiple Inheritance
Initializer Lists	Comparing Reference Types	Virtual Derivation
Range Based for Loop	Rvalue Reference Usage	Polymorphism
Null Pointer Constant	Assignment Operator	Virtual Functions
constexpr Keyword	Copy Constructor	Abstract Classes
Static Asserts	Passing and Returning Objects	Interfaces in C++
Lambda Functions	Passing References to Objects	Destructors and Inheritance
Strongly Types Enums	Move Constructor	Virtual Destructors
User Defined Literals	Move Assignment Operator	override Specifier
Raw String Literals	Golden Rule of 5	final Specifier
Module 4 : Smart Pointers	**Module 5 : Operator Overloading**	**Module 6 : Templates**
unique_ptr	Syntax Operator Overloading	Template Functions
Using unique_ptr	Overloading Numeric Types	Template Specialization
Specialization for Arrays	Overloading Overview	Template Parameter List
Replacement for std::auto_ptr	Overloading Restrictions	Inclusion Compilation Model
std::make_unique	When not to Overload	Class Templates
shared _ptr Pointer	Operators as Class Members	Template Member Functions
Control Block	Operators as Friend Functions	Template Parameter Scope
shared_ptr Destruction Policy	Overloading Stream Operators	Templates and Statics
shared_ptr Interface	Overloading ostream and istream	Templates and Friends
Cyclic References	Overloading Unary Operators	Alias Templates
weak_ptr	Overloading Binary Operators	Perfect Forwarding
Module 7 : Exception Handling	**Module 8 : Multiple Threads**	**Module 9 : Synchronization**
Error Conditions and Exceptions	Multiple Threads	Data Corruption
Class Objects as Exceptions	Benefits and Drawbacks	Lock Guard
Parameter Catch Block	Thread Class	Automatic Lock Management
Catching in Hierarchy	Joining Threads	Mutex and RAII
Golden Rule	Detaching Threads	Recursive Locking
Rethrowing Exceptions	Thread ID	Atomic Types
noexcept Specifier	Callables	Call Once
Preventing Resource Leaks	Passing Parameters	Event Handling
RAII Idiom	Pass by Reference	Condition Variables
C++ Standard Exceptions	Pass by std::ref and std::move	Wait and Notify
User Defined Exceptions	Member Function as Thread	Promises and Futures
Exception Handling Costs	Thread Local Storage	Asynchronous Tasks
Module 10 : C++14-17-20 Features	**Module 11 : Standard Template Library**	
Init-statement for if	STL Core Components	
Selection Initialization	Containers, Algorithms and Iterators	
Structured Binding Declarations	Vectors, Lists and Dequeues	
const if Expressions	Adapters and Associative Containers	
Guaranteed Copy Elision	Maps and Hash Maps	
Inline Variables	Bitsets	
Fold Expressions	STL Iterators	
Optional Type	Reverse and Iostream iterators	
Small String Allocations	Function objects	
String View	STL Algorithms	
Generic lambdas	Predicates and Comparators	
Aggregate initialization	STL Allocators	

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