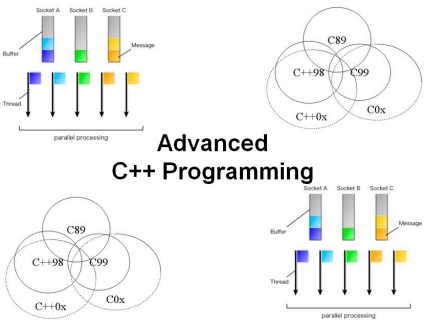
Advanced C++ Programming

Duration: 4 days **Price: € 2850**

Open Schedule



Audience Course Advanced C++ Programming

This course is intended for developers who are familiar with C++ but who want to delve into the more advanced techniques and possibilities of the language.

Prerequisites Course Advanced C++ Programming

To participate in this course knowledge of the basic concepts of C++ and extensive experience with programming in C++ is required. The participants must be familiar with topics such as operator overloading, templates, virtual functions and polymorphism.

Realization Training Advanced C++ Programming

The theory is treated on the basis of presentation slides. Illustrative demos clarify the concepts discussed. Theory and practice are interchanged and there is ample opportunity to practice.

Official Certificate Advanced C++ Programming

Participants receive an official Advanced C++ Programming certificate after successful completion of the course.

Content Course Advanced C++ Programming

In the course Advanced C++ the new and advanced aspects of the C++ language based on the standards C++11, C++14, C++17 and C++20 are extensively discussed.

C++11 Features

The course starts with an overview of the features introduced in C++11 such as type inference, initializer lists, range based for loop, lambda functions and strongly typed enums.

Right References en Move Constructors

Next, the right references and the performance gains that can be achieved by using move constructors in addition to copy constructors are discussed.

Inheritance Additions

The new possibilities regarding inheritance with the keywords default, delete, override and final are also covered. The implementation of virtual functions and the need for virtual destructors are discussed as well.

Smart Pointers

In addition, smart pointers are looked at in detail and unique pointers, shared pointers and weak pointers are treated.

Operator Overloading and Templates

Operator overloading and templates are also on the program, which focuses on variadic templates and perfect forwarding.

RAII Pattern

The modern C++ Resource Acquisition is Initialization or RAII idiom is discussed with exception handling.

Multithreading

Threads as well as the synchronization between threads are part of the standard and are discussed. This also includes asynchronous calls with promises and futures.

C++11, C++17 and C++20 Features

Then specific C++11, C++17 and C++20 features are discussed such as optional types, structured binding declarations and constructions from the world of functional programming such as fold expressions.

Advanced STL

The course concludes with an overview of advanced options of the 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

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

Modules Course Advanced C++ Programming

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

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