Stand 16.05.2025

Modernes C++: Die wichtigsten Sprachneuerungen von C++11 bis C++20 - Live-Online-Training

**Ziele -
Ihr Nutzen**

In diesem C++ Kurs lernen Sie alle wesentlichen neuen Features von C++11 und C++14 kennen, damit Sie diese künftig zum Schreiben von effizienterem und leichter wartbarem Code einsetzen können.

Enthalten ist (im Umfang von ca. 1,5 Tagen) eine Einführung in die grundlegenden Prinzipien der Compilezeit-Metaprogrammierung mit Templates, welche durch die "Variadischen Templates" (eingeführt mit C++11) an Mächtigkeit gewonnen hat, deren Implementierung sich wiederum über "Fold Expressions" (geplant für C++1z) deutlich vereinfacht.

Teilnehmer

Dieser C++ Kurs richtet sich an erfahrene Software-Entwickler mit guten C++-Kenntnissen, welche ihr Know-how gezielt ausbauen wollen in Bezug auf die mit den ISO-Standards von 2011 und 2014 eingeführten Neuerungen im Bereich der Sprach-Syntax, der Standard-Bibliothek und der Template-Programmierung.

Voraussetzungen

Umfangreiche und durch entsprechende Praxis untermauerte Erfahrungen mit C++98. Hinweis: Dieses Training eignet sich NICHT zur Auffrischung von Grundkenntnissen oder seit längerer Zeit nicht (mehr) praktisch angewandter C++-Kenntnisse.

Live Online Training

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-C++/MOD

Präsenz-Training - Deutsch

Termin	Dauer
15.09. – 18.09.2025	4 Tage

Modernes C++: Die wichtigsten Sprachneuerungen von C++11 bis C++20 - Live-Online-Training

Inhalt

KLEINE ERWEITERUNGEN

"nullptr", "constexpr", "static_assert"

Neue Formen von String-Literalen

Benutzerdefinierte Literal-Suffixe

Standardisierte Literal-Suffixe [C++14]

Vereinheitlichte Initialisierung

"std::initializer_list"

Neue Return-Typ Syntax

MICRO CONSULT

Stand 16.05.2025

Bereichsbasierte for-Schleifen

Alias-Typdefinitionen

"enum"-Klassen

ERWEITERTE TYP-DEDUKTION

Rvalue-Referenzen und Move-Semantik

auto-Variablen

auto-Funktionsergebnisse [C++14]

"decltype" und "decltype(auto)" [C++14]

LAMBDAS

Grundlagen und Prinzipien

Capture-Listen (Closures)

"Init-Captures" und "Generic Lambdas" [C++14]

Vor- und Nachteile im Vergleich mit

- Funktionszeigern
- Funktoren
- "std::bind"

Kompatibilität zu "std::function"

ERWEITRUNGEN BEI KLASSEN

Sperren und Erlauben von Defaults

Direkte Member-Initialisierung

Konstruktor-Delegation und -Vererbung

"final" und "override"

ERWEITERUNGEN DER STANDARD-BIBLIOTHEK

ash-basierte Container

Erweiterungen bei

- "std::string"
- Containern
- Algorithmen

"std::tuple"

Reguläre Ausdrücke

Zeiten und Zeitpunkte ("std::chrono")

"SMART-POINTER"

std::unique_ptr" (unique ownership)

"std::shared_ptr" (shared ownership)

"std::weak_ptr" (temporary ownership)

Problematik zyklischer Referenzen

Nutzung als RAII-Wrapper ("Custom-Deleter")

Performance und Memory-Footprint

PARALLELISIERUNG VON ABLÄUFEN

"std::async" und "std::future"

Synchronisation mit "std::mutex"

RAII-Stil Wrapper "std::lock"

"Readers/Writer-Locks" [C++14]

"Condition Variables"

Lock-freie Algorithmen

Explizites Thread-Handling

TEMPLATES UND COMPILEZEIT-METAPRGRAMMIERUNG

"constexpr"-Funktionen [C++11 vs. C++14]

"Perfect Forwarding"

Standardisierte Type-Traits

"noexcept" (Spezifikation und Compilezeit-Test)

Variadische Templates

- Grundlegende Syntax
- Expansion von Parameter-Packs

Compilezeit-Metaprogrammierung

"Fold-Expressions" [C++1z]

SONSTIGE C++11/14/1z NEUERUNGEN

Generelles Syntax "Clean-Up"

"alignas" und "alignof"

Garbage-Collection API

"Attributes" (inkl. Standard-Attribute)

"Concepts Light" [C++1z]

Begleitend: Mikro-Projekte

Demo-Code und/oder Aufgaben zur eigenen Bearbeitung nach Wahl

Inkl. anschließender Erläuterung möglicher Variationen