MICRO CONSULT

Stand 08.01.2026

C++ Schulung für Fortgeschrittene: Aufbauwissen für C++ Entwicklerinnen und Entwickler - Live-Online-Training

Mit steigender Softwarekomplexität ist es in vielen Applikationen sinnvoll, fortgeschrittene C++ Konstrukte einzusetzen; gleichermaßen unterstützen Änderungen und Erweiterungen des aktuellen C++ Standards.

**Ziele -
Ihr Nutzen**

Sie erwerben fortgeschrittene Kenntnisse im Bereich der klassischen objektorientierten Programmierung mit C++, der Nutzung von Templates und der Standardbibliothek. Ferner erhalten Sie einen Überblick zu Exceptions, Metaprogrammierung und Multithreading. Unter alternativen Lösungsansätzen können Sie informiert auswählen und dabei Laufzeit-Performance und Ressourcen-Verbrauch angemessen berücksichtigen.

Teilnehmer

Der C++ Kurs für Fortgeschrittene richtet sich an Programmierer/innen, Software-Entwickler/innen, Software-Designer/innen und Software-Architekt/innen.

Voraussetzungen

Sie sollten die C++ Grundlagen, wie sie im Training "C++ für Ein- und Umsteiger" vermittelt werden, beherrschen.

Live Online Training

26.01. – 29.01.2026	2.600,00 € *	4 Tage

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-C++/FOR

Präsenz-Training - Deutsch

Termin	Dauer
20.04. – 23.04.2026	4 Tage

Präsenz-Training - Englisch

Dauer
4 Tage

C++ Schulung für Fortgeschrittene: Aufbauwissen für C++ Entwicklerinnen und Entwickler - Live-Online-Training

Inhalt

Offener Themenblock

- Wiederholung von C++-Grundlagen (nur auf Wunsch)

Klassen, Objekte und Klassenrelationen (Vertiefung)

- Kapselung und Zugriffsschutz
- Assoziation, Aggregation und Vererbung
- Liskovsches Ersetzungsprinzip (LSP)

- Spätes Binden, abstrakte Basisklassen
- "Non Virtual Interface" -Idiom (NVI)
- Interfaces und deren Implementierung
- Besonderheiten der Mehrfachvererbung

Generische Programmierung

- Typen als Template-Parameter
- Werte als Template-Parameter
- Variationen zum Open-Close-Prinzip
- "Curiously Recurring Template"-Idiom (CRTP)
- Praktische Probleme mit Fehler-Kaskaden

Exceptions werfen und abfangen

- Grundlegendes Prinzip (auf Wunsch kurze Wiederholung)
- Best Practice für die Verwendung von "throw" und "catch"
- Standard-Exceptions als Basisklasse
- Exceptions und "std::terminate"
- Wann "noexcept" und wann nicht?
- "std::move" und "std::move_if_noexcept"

Typ-Kontrolle und -Umwandlungen

- Vor- und Nachteile statischer Typisierung
- Typ-abhängige Auswahl von Operationen
- Schlüsselwort basierte Cast-Syntax
- Statische und Dynamische Umwandlungen
- Compilezeit-Typ vs. Laufzeit-Typ und RTTI
- Typ-Anpassung durch Konstruktor
- Typ-Anpassung durch "type-cast"-Funktion
- Typsichere Operator-Überladungen

Aufrufbarer Code (Callables)

- Funktionen in C und C++
- Argumente und Rückgabewerte (teils Wiederholung)
- Funktionszeiger und Member-Funktionszeiger
- Aufrufbare Objekte (aka. Funktoren)
- Lambda-Funktionen und "Capture"-Listen
- "Type erasure" mittels "std::function"
- "std::bind" durch Lambda ersetzen
- Realisierung von Callbacks über Interfaces

Dynamisch allokierter Speicher

- Zeiger vs. Referenzen (auf Wunsch kurze Wiederholung)
- Direkte Verwendung von "new" und "delete"
- "std::unique_ptr" (exklusive Eigentümerschaft)
- "std::shared_ptr" (geteilte Eigentümerschaft)
- "std::shared_from_this" (Eigentümerkreis erweitern)
- "std::weak_ptr" (fremde Eigentümerschaft)
- Custom Deleter für "std::unique_ptr"
- Custom Deleter für "std::shared_ptr"
- Typ-Umwandlungen zwischen Smart-Pointern

Standard Template Library

- Kurze Wiederholung zum STL-Design (nur auf Wunsch)
- Flexibilität durch Trennung von Containern und Iteratoren
- Algorithmen-Überblick (C++98, C++11, C++14 …)
- "std::tuple" und "Structured Binding"
- "std::optional", std::any", "std::variant"

Praktische Metaprogrammierung

- Laufzeit-Code vs. Compilezeit-Code
- "constexpr", "consteval" und "constinit"
- Templates als Compilezeit-Funktionen
- Standard Type-Trait Templates
- Type-Trait basierte Argumenttypen
- Type-Trait basiertes Überladen

- C++20 Konzepte

Multithreading

- Kurze Einführung zu Threads (nur auf Wunsch)
- Asynchrone Funktionsaufrufe
- Threadübergreifendes Exception Handling
- Grundlegende Bausteine "Promise" und "Future"
- Thread-Pools mit "Packaged Task"
- Synchronisierung mittels Mutex
- Deadlocks und Race-Conditions
- Mutex sicher entsperren (RAII-Idiom)

Atomic Datentypen

- Erweiterte Verhaltensgarantien
- Atomares Setzen und Testen
- Nutzung Lock-freier Algorithmen
- Race-Conditions bei Smart-Pointer
- Automatisch synchronisierte Streams

Gewichtung nach Teilnehmer-Interesse

- Bedingt durch den Stoffumfang der Sprache C++ und der zugehörigen Standardbibliothek können nicht alle Themengebiete in gleicher Ausführlichkeit behandelt werden. Da alle Punkte aber zumindest einmal angesprochen werden, kann der Teilnehmerkreis insgesamt die weitere Behandlung steuern. Sofern für einzelne Punkte ein besonderes Interesse erkennbar ist, werden diese gezielt vertieft.

Praktische Übungen

- Das neu gewonnene Wissen wird in praktischen Programmierübungen vertieft.
- Der Umfang und die Anzahl der Übungen richten sich nach dem Interesse der Teilnehmenden.

Microconsult Plus:

- Sie erhalten von uns Ihre Übungsverzeichnisse und Lösungsbeispiele für alle Übungsaufgaben.
- Sie bekommen alle C++ Beispiele in elektronischer, kompilierbarer Form und können diese sehr einfach für Ihr Entwicklungsenvironment anpassen.
- In Präsenzkursen stehen in den MicroConsult-Räumen PCs an den Arbeitsplätzen zur Verfügung. Für Online-Trainings und Onsite-Schulungen wird ein Remote-Zugang zur eingerichteten Übungsumgebung bereitgestellt. Wer möchte, kann aber auch einen eigenen Laptop mit vertrauter Entwicklungsumgebung (mindestens C++20) verwenden.

HINWEIS: Die Kursunterlagen sind auf Englisch