Stand 29.03.2024

C++ Schulung für Fortgeschrittene: Weiterführende Programmiertechniken für C++ 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 wenden Templates und Exceptions effizient an. Sie haben fundierte Kenntnisse der Standard Template Library (STL). Sie realisieren fortgeschrittene objektorientierte Konzepte und Designs mit C++, auch zusammen mit C++ Multithreading. Sie sind in der Lage, bestehende und neue Applikationen in Bezug auf Performance und Verbrauchsverhalten zu optimieren.

Teilnehmer

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

Voraussetzungen

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

Live Online Training

01.07. – 04.07.2024	2.600,00 €	4 Tage
17.02. – 20.02.2025	2.600,00 €	4 Tage

* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-C++/FOR

Präsenz-Training - Deutsch

Termin	Dauer
22.04. – 25.04.2024	4 Tage
21.10. – 24.10.2024	4 Tage

Präsenz-Training - Englisch

Dauer
4 Tage

C++ Schulung für Fortgeschrittene: Weiterführende Programmiertechniken für C++ Entwickler - Live-Online-Training

Inhalt

Themeneinleitung

- Historie zu C++
- Prinzipielle Compiler-Funktionalität
- Praxistipps: Wertvolle Links im Internet

Kurze Zusammenfassung der C++ Grundlagen

- Variablen-Kategorien, Typen, Alignment
- Klassen und Objekte
- Konstruktor-Arten und Destruktor
- Operatoren mit Überladung
- Funktionszeiger in Klassen
- Strings und Streams
- Klassenrelationen: Assoziation, Selbst-Assoziation, Aggregation, Komposition, Vererbung, Mehrfachvererbung und Alternativen
- Interface-Konzept mit rein virtuellen Funktionen
- Neue Features der aktuellen Standards
- Übung: Sie verstehen die vorgegebene SW-Architektur und lernen dabei das Builder-Pattern kennen, implementieren Klassen, Komposition und Vererbung und testen diese automatisiert nach dem TDD-Vorgehen (Test Driven Development).
- Dabei berücksichtigen Sie Qualitätsaspekte wie objektorientierte Programmierung, Modularisierung, Wiederverwendbarkeit und Erweiterbarkeit

Exceptions

- Erläuterung und Programmierung Exception Handling
- Exception-Klassen und -Hierarchien
- Benutzer-Exceptions
- C++ Standard-Exceptions
- Praxistipps: Konzepte, Richtlinien
- Übung: Sie erweitern die Übungsapplikation um flexible
- Ausnahmebehandlung mit Exceptions

New Style Casts

- Static, dynamic, const und reinterpret Cast
- Die richtige Wahl beim Einsatz
- Bezug zu RTTI und Exception Handling

Runtime Type Information (RTTI)

- Erläuterung und Programmierung von RTTI
- Klasse type_info
- Verwendungsmöglichkeiten und Konsequenzen beim Einsatz

Lamda-Funktionen

- Syntax und Anwendung
- Closures
- std::function
- Binden von Parametern

Speichermanagement

- Speichersegmente (BSS Block Started by Symbol, Heap, Stack)
- Vergleich und Bewertung der Datensegmente
- Dynamisches Speichermanagement mit new und delete
- Überladen (lokal und global) von new und delete
- Algorithmen
- Virtueller Destruktor
- Placement new
- Bezug zu Exception Handling
- Smart Pointer: unique_ptr, shared_ptr, weak_ptr
- Casten von Smart-Pointern
- Praxistipps: Risiken erkennen und Stolpersteine vermeiden

Template-Funktionen und Template-Klassen

- Prinzipielle Funktionsweise
- Template-Funktionen, Template-Klassen und deren Anwendung
- Beispiele für Template-Klassen
- Vererbung und Interfaces mit Template-Klassen
- Container/Algorithmen im STL-Stil
- Laufzeit- vs. Compilezeit-Polymorphismus
- Type-Traits
- Perfect Forwarding mit Templates

- Variadic Template-Funktionen und Template-Klassen
- Alias-Templates
- Praxisbeispiele für Template-Klassen
- Übung: Sie wenden das Observer-Pattern im Design der Applikation an und implementieren es basierend auf einer containerartigen eigenen Template-Klasse

STL Standard Template Library

- Container, Container-Adapter
- Iteratoren
- Algorithmen, Funktionsobjekte
- Speicher-Allocator-Klasse
- Praxistipp: Übersicht über alle STL-Containerelemente und deren Zusammenhänge
- Übung: Sie wenden das Observer-Pattern im Design der Applikation an und implementieren es basierend auf einer STL-Container-Klasse

Multithreading und Atomic-Datentypen

- Multithreading-Grundkonzepte
- Threads, Mutex, Condition Variable, Future
- Anwendung der Mechanismen
- Übung: Sie adaptieren die Applikation an einen Timer und steuern sie über einen Thread. Dabei nutzen Sie mit ihren Vorteilen eine zusätzliche Betriebssystem-Abstraktion mit Wrapper-Klassen.

Typische Fallstricke und verbreitete Idiome (PIMPL, RAII, NVI, ...)

- RAII (Resource Acquisition Is Initialization), Ressourcen-Wrapper
- NVI (Non-virtual Interfaces)
- PIMPL (Pointer to Implementation)
- Handling Self-Assignment in Assignment Operator
- Praxistipps zu weiteren C++ Idiomen

Übungen im C++ Kurs für Fortgeschrittene

- Für die Implementierung der durchgängigen Übung (Uhrenapplikation) verwenden Sie das Microsoft Visual Studio.

MicroConsult PLUS:

- Sie erhalten von uns Ihre Übungsverzeichnisse und Lösungsbeispiele für alle Übungsaufgaben.