

## C++: Standard- und Boost-Library Workshop - Live-Online-Training

### Ziele - Ihr Nutzen

Neben der sicheren Beherrschung der Sprachsyntax werden von C++-EntwicklerInnen solide Kenntnisse im Bereich der Standard-Bibliothek erwartet, um deren Funktionalität im Rahmen der zu lösenden Aufgaben unterstützend einzusetzen.

Da die Standard-Bibliothek der ersten Standardisierung (mit C++98) durch die Sprachstandards (C++11 und C++14) auf den nahezu doppelten Umfang gewachsen ist, wurde der STL-Teil ausgelagert (abgesehen von einem sehr kurzen Überblick), während das hier beschriebene Training alle anderen Teile behandelt (siehe auch Inhaltsangabe).

In Bereichen, in denen nach aktuellem C++ Standard noch Lücken bestehen, werden auch auf der Boost-Plattform entstandene Library-Komponenten einbezogen, ebenso wenn Boost-Komponenten vorhandene Standard-Funktionalitäten in besonders praxisrelevanter Weise erweitern.

### Teilnehmer

Software-Entwickler, Software-Entwicklungsleiter, Software-Architekten.

### Voraussetzungen

Grundlegende Programmierkenntnisse in C++.

### Live Online Training

\* Preis je Teilnehmer, in Euro zzgl. USt.

Anmeldecode: L-C++/LIB

### Präsenz-Training - Deutsch

#### Dauer

4 Tage

## C++: Standard- und Boost-Library Workshop - Live-Online-Training

### Inhalt

#### DURCH C++11/14 ERSETZE BOOST-FEATURES

##### Container-Initialisierung

- C++11 "std::list\_initializer"
- vs. Boost.Assign

##### Sperrungen von Defaults

- C++11 "= delete"
- vs. Boost.NonCopyable

##### Bereichsbasierte Schleifen

- C++11: Range-"for"
- vs. BOOST\_FOREACH

**Lambdas (Funktions-Literale)**

- C++11 Lambda-Syntax
- vs. Boost.Lambda

**Statische Zusicherungen**

- C++11 "static\_assert"
- vs. BOOST\_STATIC\_ASSERT

**Binäre Konstanten**

- C++14 "0bXXXX"
- vs. BOOST\_BINARY

**"Move-Only"-Datentypen**

- C++11 Rvalue-Referenzen
- Boost.Scoped\_ptr
- Boost.Ptr\_container

**KLEINERE HELFER****Umwandlung zwischen Strings / Zahlen**

- C++11 "std::to\_string" / "std::strtoXX"
- Boost "lexical\_cast"
- Boost "numeric\_cast"

**C++14 Standard-Literal-Suffixe****Boost.ScopeExit****Boost.ProgramOptions****ZEICHENKETTEN-VERARBEITUNG****Zeichen-Grundtypen und -Kodierung****"std::string" vs. "std::vector"****Reguläre Ausdrücke für**

- Vergleiche
- Zerlegungen
- Ersetzungen

**Boost.String\_algo****Boost.Tokenizer****CONTAINER, ALGORITHMEN UND ITERATOREN****Kurzer STL-Überblick****Auswahlkriterien für Container**

- STL-Container (inkl. "unordered associative")
- Boost-Container (BiMap, MultiIndex, PropertyTree)

**Boost-Erweiterungen bei**

- Algorithmen (insbesondere Boost.Range)
- Iteratoren (insbesondere Boost.Iterator)

**Tuple****Boost.Optional****Boost.Any****Boost.Variant****EIN-/AUSGABE (STREAMS)****Grundlegendes Design****Erweiterbarkeit**

- bei E/A-Operatoren
- beim Buffer-Interface

**Boost.ioStateSavers**

**Boost.Format**

**Boost.ioStreams**

**Boost.Serialization**

**Boost.FileSystem**

## **FUNKTIONALE PROGRAMMIERUNG**

**Funktionen, Funktoren und Lambdas ("Callables")**

**Lambdas und "Closures"**

**Lambdas vs. "std::bind"**

**Type-Erasure mit "std::function"**

**Boost.Bind und Boost.Function**

## **SMART-POINTER**

**C++11 "std::unique\_ptr"**

**C++11 "std::shared\_ptr"**

**C++11 "std::weak\_ptr"**

**Boost.IntrusivePtr**

## **CHRONO-LIBRARY**

**Grundlegende Konzepte**

**Zeitpunkte ("clocks")**

**Zeitspannen ("durations")**

**Boost.Chrono**

## **RANDOM-LIBRARY**

**Vor-/Nachteile von "std::rand" (C89)**

**C++11 Generatoren für Zufallszahlen**

**C++11 Zufalls-Verteilungen**

**Boost.Random**

## **PARALLELISIERUNG VON ABLÄUFEN**

**C++11 Abstraktionen**

- asynchroner Funktionsaufruf
- "Mutex"- und "Lock"-Synchronisation
- "Condition Variables"

**C++11 Explizites Multithreading**

**Boost.Thread**

## **COMPILEZEIT-METAPROGRAMMIERUNG**

**C++11 Standardisierte Type-Traits**

- Compile-Zeit Tests ...
- ... und Typ-Berechnungen
- C++14 Vereinfachungen

**C++11 "std::enable\_if"**

**C++11 "std::ratio"**

**C++1z "Fold Expressions"**

**Boost.Calltraits**

**Boost.MPL****Überblick zu weiteren Boost-Libraries**

- Boost.Asio
- Boost.Exception
- Boost.Flightweight
- Boost.Fusion
- Boost.Interval
- Boost.Msm und Boost.Statechart
- Boost.Preprocessor
- Boost.PropertyMap
- Boost.Proto,
- Boost.Rational
- Boost.Signals / Boost.Signals2
- Boost.Spirit (Qi, Karma, Phoenix)
- Boost.Test
- Boost.Tribool
- Boost.Units
- Boost.Uuid
- Boost.Wave
- Boost.Xpressive

**Begleitend: Mikro-Projekte**

- Demo-Code und/oder Aufgaben zur eigenen Bearbeitung nach Wahl
- inkl. anschließender Erläuterung möglicher Variationen