MICRO CONSULT

As of 14.01.2026

C++ Advanced Training: Extended Knowledge for C++ Developers - Face-to-Face Training

In view of the increase in software complexity, many applications benefit from advanced C++ constructs; additional support is provided by modifications and extensions of the current C++ standard.

Objectives

You will acquire advanced knowledge of classic object-oriented programming with C++, the use of templates, and the standard library. You will also get an overview of exceptions, metaprogramming and multithreading. You will be able to make an informed choice from alternative approaches based on runtime performance and resource consumption.

Participants

The C++ advanced training addresses programmers, software developers, software designers and software architects.

Requirements

Solid knowledge of C++ basics as covered in our training "C++: Object-Oriented Programming"

C++ Advanced Training: Extended Knowledge for C++ Developers - Face-to-Face Training

Content

Open Topic Block

- Recap of C++ basics (only if required)

Classes, Objects and Class Relations (in-depth)

- Encapsulation and access protection
- Association, aggregation and inheritance
- Liskov's substitution principle (LSP)
- Late binding, abstract base classes
- "Non virtual interface" idiom (NVI)
- Interfaces and their implementation
- Specifics of multiple inheritance

Generic Programming

- Types as template parameters
- Values as template parameters
- Variations of the open-close principle
- "Curiously recurring template" (CRTP) idiom
- Problems with error cascades in practical application

Throwing and Catching exceptions

- Basic principle (short recap if required)
- Best practice for the use of "throw" and "catch"
- Standard exceptions as a base class
- Exceptions and "std::terminate"
- When to use "noexcept"
- "std::move" and "std::move_if_noexcept"

Type Control and Conversions

- Advantages and disadvantages of static typing

- Type-dependent selection of operations
- Keyword-based cast syntax
- Static and dynamic conversions
- Compile time type vs. runtime type and RTTI
- Type customization through constructor
- Type customization through "type cast" function
- Type-safe operator overloads

Callable Code (Callables)

- Functions in C and C++
- Arguments and return values (in part by recap)
- Function pointers and member function pointers
- Callable objects (aka. functors)
- Lambda functions and "capture" lists
- "Type erasure" using "std::function"
- Replace "std::bind" with Lambda
- Realization of callbacks via interfaces

Dynamically Allocated Memory

- Pointer vs. references (short recap if required)
- Direct use of "new" and "delete"
- "std::unique_ptr" (exclusive ownership)
- "std::shared_ptr" (shared ownership)
- "std::shared_from_this" (extend ownership)
- "std::weak_ptr" (non-owning)
- Custom deleter for "std::unique_ptr"
- Custom deleter for "std::shared_ptr"
- Type conversions between smart pointers

Standard Template Library

- Brief recap of STL design (only if required)
- Flexibility by separating containers and iterators
- Algorithm overview (C++98, C++11, C++14, ...)
- "std::tuple" and "Structured Binding"
- "std::optional", 'std::any', "std::variant"

Practical Metaprogramming

- Runtime code vs. compile time code
- "constexpr", "consteval" and "constinit"
- Templates as compile-time functions
- Standard type-trait templates
- Type-trait based argument types
- Type-trait based overloading
- C++20 concepts

Multithreading

- Short introduction to threads (only if required request)
- Asynchronous function calls
- Exception handling across threads
- Basic building blocks "promise" and "future"
- Thread pools with "packaged task"
- Synchronization by using mutex
- Deadlocks and race conditions
- Safe unlocking of mutex (RAII idiom)

Atomic Data Types

- Extended behavior guarantees
- Atomic setting and testing
- Use of lock-free algorithms
- Race conditions for smart pointers
- Automatically synchronized streams

Topic Focus Weighting According to the Attendees' Interest

- Due to the extensive scope of the C++ language and the associated standard library, not all topics can be

covered in equal detail in one dedicated training course. However, all topics will be addressed at least once. Those topics, which are of specific interest for the attendees, will be covered in more detail.

Supervised Exercises in Workshop Style

- A practical exercise that is extended step by step completes the training course and will account for about 50 % of the training scope, in continuous blocks of 90 to 120 minutes.
- A sample solution is provided in advance for each step so that the individual subtasks can either be programmed independently or executed as a "side-by-side" review of the differences, while the remaining supervised lab time can be used to deepen your knowledge of C++ topics as required.

MicroConsult PLUS / BYOD

- All code examples covered in the training are available online in compilable format for in-depth study. For face-to-face trainings, MicroConsult provides the attendees with PCs in the classrooms. For online trainings and onsite courses, there is remote access to the training environment available. You are also welcome to use your own laptop with your familiar development environment (at least C++20).

FACE-TO-FACE TRAINING

Price *	Duration
2.600,00 €	4 days

Training code: E-C++/FOR
* Price per attendee, in Euro plus VAT

Face-To-Face - German

Date	Duration
20.04. – 23.04.2026	4 days

Live Online - German

Date	Duration
26.01. – 29.01.2026	4 days

Coaching

Our coaching services offer a major advantage: our specialists introduce their expertise and experience directly in your solution process, thus contributing to the success of your projects.

We will be happy to provide you with further information or submit a quotation tailored to your requirements.