MICRO CONSULT

As of 30.04.2026

Developing Clean, Efficient and Robust C++ Software with Classic Object Orientation and Modern C++ Language Tools - The Path to Clean Code

Objectives

Master the most essential best practices and create clean, highly performant and robust C++ code from scratch or modernize existing legacy code in the scope of a refactoring project.

Participants

Software developers, software architects

Requirements

For active participation in the lab: You already know the basics of object-oriented programming and have a solid basic knowledge of C++. You are familiar with a common development environment that supports C++20. You can use your own laptop for the exercises or work via an internet browser in an online workspace provided by us.

Optional: If you have little or no C++ programming experience, you may choose not to perform the programming exercises in the labs and rather focus on a "before-after" comparison of each step in the form of a code review.

Developing Clean, Efficient and Robust C++ Software with Classic Object Orientation and Modern C++ Language Tools - The Path to Clean Code

Content

Paradigms, Patterns, Idioms and Best Practices

- Good code from the start
- Good code through refactoring
- Possible motivation challenges
- Hidden costs of bad code
- Conflicting goals and trade-offs

Basic Rules

- Avoiding repetitions
- Openness for extensions
- Complexity limitation
- Modularization
- Test automation
- Optimization only with clear goals

C++ Specific Aspects

- Header and implementation files
- Language vs. libraries and frameworks
- Advantages and disadvantages of templates
- The role of metaprogramming

Useful Items of the Standard Library

- Brief overview of C++98 to C++23 STL - containers, iterators, algorithms
- "std::any", "std::variant"
- "std::tuple" and "structured binding"
- "std::function"

Classic Object Orientation with C++

- Encapsulation of data and functions
- Secret principle and access protection

- Inheritance, interfaces, dynamic binding
- Composition often preferable to inheritance
- "SOLID" principles and more (SLA, YAGNI, ...)

Communication Between Software Modules

- Synchronous and asynchronous methods
- Principle of loose coupling and high cohesion
- Performance increase through callbacks
- Comparison and evaluation of typical alternatives (linkers, pointers/references, interfaces, templates)

Practical Exercises in Workshop Style

- Project-based structure: The practical exercises are structured as a continuous mini project across the entire training, with each exercise building on the previous one. Each unit is about 90 to 120 minutes, and you can define your own speed depending on your individual level of C++ knowledge.
- Support and solutions: All sample solutions are provided in advance, enabling you to stay on the right track even when the exercises are challenging. The trainer is always there to help, support, guide and answer questions. Extra challenges: For advanced attendees, each step offers additional optional 'open challenges', making the practical training section exciting and challenging even for experienced programmers.

MicroConsult PLUS:

- You will receive the code and description of the workshop exercises prior to the training such that you can familiarize yourself with the steps to be processed and the open challenges.

FACE-TO-FACE TRAINING

Price *	Duration
1.950,00 €	5 days

Training code: E-OOPFC++
* Price per attendee, in Euro plus VAT

Face-To-Face - German

Duration
3 days

Live Online - German

Date	Duration
17.11. – 19.11.2026	3 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.