Advanced C++ Programming

Level: advanced

Length: 35 – 40 hours

Course objectives
- learn and exercise more complex or newer parts of C++
- use of object oriented programming to solving practical problems by using C++

What you will learn
- Stereotypes & idioms used in programming with C++
- Particular ways to implement several design patterns
- Exercise how the design is mapped to code
- Exercise soft skills of communication and presentation

Who can participate: C++ programmers who want
- to familiarize himself/herself with usually less known aspects of the language
- to exercise the object oriented programming and use of C++ for solving interesting, more complex problems

Prerequisites: practical experience and knowledge of C++ at least at medium level

Required facilities: VGA projector, white board, computers, C++ development tools. It's highly recommended using an IDE, good (free) examples are Microsoft Visual C++ Community Edition, Microsoft Visual Code or Code Blocks

Related courses: The C++ Programming Language, Object Oriented Analysis and Design, Design Patterns

Minimal bibliography: The C++ Programming Language, Fourth Edition, Bjarne Stroustrup, Addison-Wesley, ISBN 0-321-56384-0

Description

This course is targeted to C++ programmers who want to deep their knowledge about the language and ways to use it.

There are discussed issues and details related to inheritance, polymorphism, Runtime Type Information, operator overloading, templates implementation, multi-threading programming.

The training is highly interactive, the attendees are implied in discussing the ideas and in designing solutions which are ultimately expressed in C++. The main purpose of this training is to exercise object oriented programming by using C++.

Note: the subjects are adapted to the attendees' profile, their background, experience, goals and time constraints. We can approach other subjects depending on the context.

Examples of topics

1. Operator overloading: (), [], ->, smart pointers, how to implement, RAII idiom, how to use PIMPL idiom in order to isolate a changing service.

2. Object Pool implementation – plug in the service by overloading of new & delete operators.

3. Inheritance, polymorphism, virtual methods, implementation of polymorphism, implications of multiple inheritance on code generation, pointer & reference casts, applications. Methods of solving problems in OOP.

4. Run-time Type Information – typeid operator, type_info class, dynamic_cast, C++ casts, how to give support to conversions that imply objects, implicit and explicit conversions.

5. Memory management in C++; new/delete variants, placement new application to iimplement a proxy object, small object allocators

6. Programming with templates - partial specializations, explicit specialization, variadic templates, policy based programming. Technics for extending the functionality at compile time based on templates. Code generation implications. Static polymorphism by using the CRTP idiom. Dynamic & static composition – as an example of how to solve problems runtime or at compile time.

7. Standard Template Library: algorithms, function objects (functors), predicates, callables in C++. Use of lambda functions, how they are implemented.

8. Modern C++, new features of C++11 and newer versions – uniform initialization, auto, smart pointers, scoped enumeration, explicit, for each, rvalue references, move semantics, special class members, rule of five, defaulted and deleted functions, lambda functions, variadic templates.

9. Concurrent programming with threads, specific problems – access to common resources & threads collaboration, threads ecosystem, use of mutex & condition variable, concurrency patterns.