

Training program:

Advanced Modern C++

Info:

Name:	Advanced Modern C++
Code:	adv-modern
Category:	C and C++
Target audience:	architects developers
Duration:	3-6days
Format:	40% wykład / 60% workshop

Training program

1. C++ Basics for Experts

1.1. Advanced approach to C++ basic concepts (identifiers, types and their properties, objects, scope, lifetime, and more)

1.2. Class special member functions undercover

1.3. Object construction and initialization voodoo

2. Coding with performance in mind

2.1. Value categories

2.2. Copy elision

2.3. Move semantics

2.4. Ref-qualifiers

2.5. noexcept

2.6. constexpr

2.7. Source code vs hardware - introduction

3. Utilities that every C++ developer should know and use

3.1. Smart pointers

3.2. Lambda expression

3.3. Algorithms and their specializations

3.4. `std::string_view`

3.5. `std::optional`

3.6. `std::tuple`

3.7. `std::variant`

4. Templates demystified

4.1. Class, function, variable, and alias templates

4.2. Parameters and arguments

4.3. Explicit and partial specialization

4.4. Explicit and implicit instantiation

4.5. Template argument deduction

4.6. Variadic templates

4.7. Fold-expressions

4.8. Dependent names

4.9. SFINAE

5. Tools mandatory in modern C++ developer's workshop

5.1. cmake

5.2. clang toolset

5.3. gtest/gmock

5.4. Code Coverage

5.5. Google benchmark

5.6. Compiler Explorer

6. Design patterns do not end on GoF

6.1. Non-Copyable

6.2. RAII

6.3. Copy-and-swap

6.4. Smart Pointer

6.5. Type Traits

6.6. Tag dispatch

6.7. Policy-based design

6.8. EBO

6.9. Type Erasure

6.10. Copy-on-write

6.11. CRTP

6.12. Singleton

6.13. SOO

7. C++ is not only about OOD

7.1. Inheritance vs polymorphism

7.2. Pitfalls and how to fight with them

7.3. Value Semantics

7.4. Concept-based Polymorphism

8. C++ containers for demanding developers

8.1. Finding the right tool for the job (functionality, memory layout, performance/latency)

8.2. Sequence containers

8.3. Associative containers

8.4. Hash tables

8.5. Not only from C++ standard library

9. Writing stable and secure code

9.1. Typical problems of legacy code

9.2. Pitfalls of buggy coding standards

9.3. Class construction and destruction

9.4. Magic numbers everywhere

9.5. Replacing pointers with C++ constructs

9.6. Value objects

9.7. Error handling