Sela.

BoostCPP

Boost the Modern C++

college@sela.co.il 03-6176666

Boost the Modern C++

BoostCPP - Version: 1

3 days Course

Description:

Let's look at the big picture and discuss what C++ developers do in 2023 to achieve their goals. What your individual goals are and use the experience and expertise of everyone to help you become a better C++ developer! Starting with exploring the big picture and look at what goals C++ developers try to accomplish. Learning tools, techniques C++ developers use to reach those goals, their trade-offs and what the C++ community considers state-of-the-art. Diving deeper to break up the big picture into details, finding out where you stand and can easily identify what would help you to solve the kind of problems you face in your daily work as a C++ developer. Looking specifically at those areas in C++ which will help you most.

Intended audience:

C++ developers who have exhausted the standard library and search for new ways to increase their productivity.C++ developers who have exhausted the standard library and search for new ways to increase their productivity.C++ developers who have exhausted the standard library and search for new ways to increase their productivity.C++ developers who have exhausted the standard library and search for new ways to increase their productivity

Prerequisites:

Objectives:

Topics:

C++ hierarchy of needs

• Understanding the C++ language and its environment, including the core language features and the Standard Library, as well as the available external libraries.

Writing bug-free code

• Learning the fundamentals of debugging C++ code and writing code that is easier to debug and maintain.

Optimizing performance

• Identifying and resolving performance issues in C++ code, including memory management, optimization, and parallelization.

Time-efficient feature implementations

• Learning approaches for implementing features in a timely manner, such as agile development, unit testing, and refactoring.

Modifying and extending hard-to-change legacy code

• Learning methods for modifying and extending existing code, including reverse engineering, debugging, and code refactoring.

Object-oriented programming vs. data-oriented design

• Understanding the differences between object-oriented programming and data-oriented design, and the benefits and drawbacks of each.

Runtime vs. compile-time computations

• Learning the differences between runtime and compile-time computations, and when to use each.

Asynchronous I/O vs. multithreading

• Learning the differences between asynchronous I/O and multithreading, and when to use each.

Goals without trade-offs

• Understanding how to achieve goals without making trade-offs, such as making code clean, maintainable, and efficient.

Dependency injection

• Learning how to use the dependency injection pattern to improve code maintainability.

º Hidden champions: Useful general-purpose libraries hardly anybody knows

º Modern C++ vs. traditional C++

º Best practices