

Programming in C++

2-day “intensive” outline for beginners

FA-0368 · 2-day

Software Development

Introduction

This 2-day course in C++ programming is a specially curated program designed to instill both theoretical understanding and practical skills among participants. We're excited to announce that this course is conducted by professionals who have carved a niche for themselves in the international arena, having worked with prestigious Research and Development institutions across the globe. Their extensive industry experience translates into a curriculum that reflects real-world needs and practices, ensuring that you're not just learning C++ but understanding how it is integrated and implemented in significant, cutting-edge technological developments.

This compact course is the culmination of our trainers' expertise, bringing to the table a learning experience that extends beyond conventional classroom teaching. It offers insights into how C++ programming operates at the core of tech innovation, providing participants with the confidence and skills required to navigate the world of software development with finesse.

Learning outcomes

By the end of this 2-day journey, participants will:

- Comprehend the fundamentals and unique features of C++ in contrast to other programming languages.
- Understand and utilize the syntax and semantics of C++ to write efficient and effective code.
- Explore object-oriented programming in C++, understanding classes, objects, inheritance, polymorphism, and encapsulation.
- Implement various operations using standard template libraries.
- Apply memory management techniques in C++, including pointers, references, and allocation/deallocation of memory.
- Understand the handling of files and streams in C++ programming.
- Gain insights into error handling through exceptions.

- Delve into advanced C++ features, including lambda expressions, move semantics, and type inference.
- Write, debug, and optimize C++ code using professional-level best practices.
- Work on a practical project that encompasses the key aspects learned during the course, aiding in the consolidation of knowledge and skills.

Prerequisites

All participants should ensure they meet the following prerequisites before enrolling:

- Basic familiarity with using a computer system.
- Understanding of fundamental programming concepts.
- Some experience with another programming language (recommended but not mandatory).
- Prior exposure to object-oriented programming concepts will be beneficial.
- Access to a computer with a suitable development environment (preferably with a C++ IDE installed, like Visual Studio, Code::Blocks, or any preferred software).
- Enthusiasm to participate in rigorous learning sessions, both theoretical and practical.

Training outline

1. Understanding the Foundations of C++:

1. Introduction to the architecture and syntax of C++.
2. Historical significance and the modern relevance of C++.
3. Setup of the C++ development environment.
4. Writing the first C++ program: structure and output.

2. Exploring Variables and Basic Types:

1. Deep dive into data types, variables, constants, and literals.
2. Introduction to operators and basic input/output.
3. Implementing type conversions.

3. Control Structures and Functions:

1. Studying if, switch, loops, and control flow.
2. Writing and invoking functions, passing parameters, and returning values.
3. Overview of recursive functions.

4. Diving into Object-oriented Programming:

1. Understanding classes and objects in C++.
2. Implementing inheritance, polymorphism, and encapsulation.
3. Utilizing constructors and destructors.
4. Overloading functions and operators.
5. Insights into object relationships and dynamic polymorphism.

5. Data Handling with Arrays, Strings, and Pointers:

1. Managing arrays and multidimensional arrays.
2. Harnessing the power of pointers and references.
3. Manipulating strings using C++ standard library functions.

6. Advanced Features of C++:

1. Unveiling the potential of templates.
2. Understanding and implementing smart pointers.
3. Utilizing advanced features like lambda expressions, auto, and decltype.
4. An introduction to the STL (Standard Template Library).

7. File Handling and Streams:

1. Basics of file input/output operations.
2. Managing files and streams effectively.
3. Implementing file operations in practical scenarios.

8. Exception Handling:

1. Basic error-handling mechanisms.
2. Implementing try, catch, and throw operations.
3. Creating custom exceptions.

9. Effective Debugging and Testing Techniques:

1. Strategies for debugging C++ programs.
2. Writing test cases and performing testing.
3. Introduction to optimization techniques.

10. Conclusive Project:

1. Undertaking a project that encapsulates the primary concepts learned.
2. Encouraging creative and critical thinking, problem-solving, and practical application of knowledge.
3. Trainers' guidance and feedback, reflecting real-world developmental oversight.

The course's structure, though intensive, ensures a thorough understanding by interlinking theory with hands-on practices, group interactions, and individual assessments. Regular Q&A sessions and interactive problem-solving make learning dynamic, catering to individual queries and doubts. This approach ensures that every participant not only understands C++ on a fundamental level but also gains insight into its practical application in solving complex, real-world challenges.