Course Outline

Modern C++ Fundamentals

Course Summary

Description

Discover the peculiar feature points of C++ with C++ Fundamentals, and lay a solid foundation of C++ knowledge. Get a hands-on, practical introduction to low-level programming with C and C++.

C++ Fundamentals begins by introducing you to the C++ syntax. You will study the semantics of variables along with their advantages and trade-offs, and see how they can be best used to write safe and efficient code. With the help of this course, you'll be able to compile fully working C++ programs and understand how variables, references, and pointers can be used to manipulate the state of the program. You will then explore functions and classes— the features that C++ offers to organize a program—and use them to solve more complex problems. You'll also understand common pitfalls and modern best practices, especially the ones that diverge from the C++98 guideline.

As you advance through the chapters, you'll study the advantages of generic programming and write your own templates to make generic algorithms that work with any type. This C++ course will guide you in fully exploiting standard containers and understanding how to pick the appropriate container for each problem. You will even work with a variety of memory management tools in C++.By the end of the course, you'll be equipped with all that you need to know to develop robust and high performance infrastructure. By the end of this course, you will not only be able to write efficient code, but also be equipped to improve the readability, performance, and maintainability of your programs using standard algorithms.

Objectives

At the end of this course, students will be able to:

- Work with the C++ compilation model and syntaxes
- Apply best practices for writing functions and classes
- Write safe, generic, and efficient code with templates
- Explore the containers that C++ standard offers
- Discover the new paradigms introduced with C++11, C++14, and C++17
- Get to grips with the core language features of C++
- Abstract complex problems using object-oriented programming in C++

Topics

- Getting Started
- Functions
- Classes
- Generic Programming and Templates
- Standard Library Containers and Algorithms
- Object-Oriented Programming

Audience

If you're a developer looking to learn a new powerful language or are familiar with C++ but want to update your knowledge with modern paradigms of C++11, C++14, and C++17, this course is for you.

Prerequisites

To easily understand the concepts in the course, you must be familiar with the basics of programming. This course is intended for individuals having prior programming knowledge. This course is not for beginners who have chosen C++ as their first programming language.

Duration

Four days

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically

Course Outline

Modern C++ Fundamentals

Course Outline

I. Getting Started
- A. The C++ Compilation Model
- B. Built-in Data Types
- C. Pointers and References
- D. Control Flow
- E. The try-catch block
- F. Arrays

II. Functions
- A. Function Declaration and Definition
- B. Local and Global Variables
- C. Passing Arguments and Returning
- D. Working with const References or r-value References
- E. Const Parameters and Default Arguments
- F. Namespaces
- G. Function Overloading

III. Classes
- A. Declaring and Defining a Class
- B. Member Functions
- C. Constructors and Destructors
- D. Resource Acquisition Is Initialization
- E. Nested Class Declarations
- F. Copy Constructors and Assignment Operators
- G. Operator Overloading
- H. Introducing Functors

IV. Generic Programming and Templates
- A. Templates
- B. Defining Function and Class
- C. Non-Type Template Parameters
- D. Making Templates Easier to Use
- E. Being Generic in Templates
- F. Variadic Templates
- G. Writing Easy-to-Read

V. Standard Library Containers and Algorithms
- A. Sequence
- B. Associative Containers
- C. Unordered Containers
- D. Container
- E. Unconventional
- F. std::optional
- G. std::variant
- H. Iterators
- I. Algorithms Provided by the C++ Standard Template Library

VI. Object-Oriented Programming
- A. Inheritance
- B. Polymorphism
- C. Virtual Methods
- D. Interfaces in C++
- E. Dynamic Memory
- F. Safe and Easy Dynamic Memory

Due to the nature of this material, this document refers to numerous hardware and software products by their trade names. References to other companies and their products are for informational purposes only, and all trademarks are the properties of their respective companies. It is not the intent of ProTech Professional Technical Services, Inc. to use any of these names generically