

# C++ Advanced

## Scope and Sequence

Version 1.0

Developed in collaboration with  
OpenEDG C++ Institute

## Contents

Target Audience .....	3
Course Prerequisites .....	3
Certification Alignment .....	3
Course Description.....	4
Course Objectives .....	5
Equipment Requirements .....	6
Course Outline .....	6
Why C++ Advanced? .....	8

## Target Audience

*C++ Advanced (CPPA)* is specifically designed for individuals who have successfully completed the *C++ Essentials 2* course, or have equivalent intermediate knowledge in C++ programming, and are now aiming to **elevate their proficiency in C++ programming to a professional level**.

This course is particularly beneficial for:

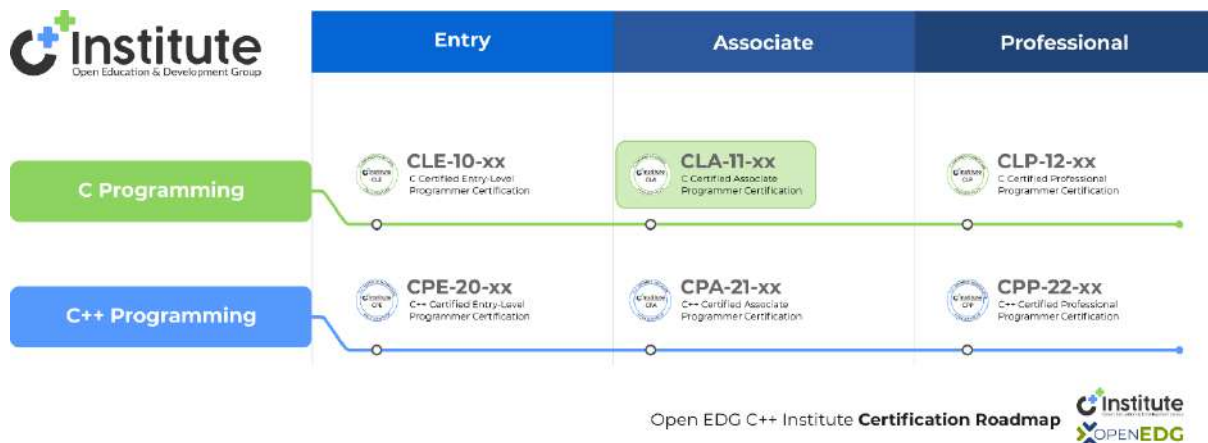
- Programmers with intermediate C++ skills aiming to delve into more complex programming concepts.
- Students and professionals targeting the [CPP – C++ Certified Professional Programmer](#) certification.
- Individuals looking to advance their careers in software development, network programming, system development, or related fields.
- Self-taught programmers seeking to deepen their C++ programming knowledge and practical skills.

## Course Prerequisites

- Successful completion of C++ Essentials course series, CPA certification, or possessing **equivalent intermediate knowledge** in C++ programming.
- **A strong understanding of intermediate programming concepts**, including functions, classes, inheritance, file operations, and error handling.

## Certification Alignment

*C++ Advanced* is your stepping stone to the [CPP – C++ Certified Professional Programmer](#) certification, serving as the direct preparatory stage for this advanced certification. Building upon the foundations laid by *C++ Essentials 1* and *C++ Essentials 2*, this course readies candidates for the CPP exam by covering advanced C++ topics such as templates, STL, algorithms, and advanced I/O, and equipping them with the necessary skills to solve complex programming problems.



The CPP certification underscores an individual's **extensive mastery in advanced C++ programming**, spotlighting key areas such as the utilization of **sequence containers** and **container adapters**, adept handling of **associative containers**, and the strategic application of **modifying** and **non-modifying algorithms**. It further examines proficiency in **sorting** and **binary searches**, **merging** and **heap operations**, along with the adept use of **STL functional objects** and **utilities**, and sophisticated **I/O operations**.

This certification validates a programmer's capability to **tackle advanced challenges using templates and modern C++ features**, positioning them as distinguished professionals in the competitive landscape of software development.

Candidates can take the *CPP – C++ Certified Professional Programmer* certification exam via the [TestNow™](#) – OpenEDG Testing Service Platform, and through the network of [Pearson VUE](#) Testing Centers.

## Course Description

Created by the OpenEDG [C++ Institute](#) and delivered in partnership with the [Cisco Networking Academy](#), *C++ Advanced* is **the natural progression for those looking to master C++ programming**.

The course dives into advanced C++ topics, preparing learners for challenging programming tasks and the CPP certification.

Through engaging **study materials**, **quizzes**, and **hands-on labs**, participants not only learn theoretical concepts but also apply them in real-world scenarios, ensuring they are ready for advanced programming roles and the CPP certification exam.

### Modules Overview:

- **Module 1: STL Sequential Containers** – Dive into sequential containers such as vector, deque, and list, learning their APIs and how to utilize sequential container adapters like stack, queue, and priority queue. Understand the best practices for managing objects within these containers and discern when each type is most appropriately used.
- **Module 2: STL Associative Containers** – Explore associative containers, including set, multiset, map, and multimap. Master their behaviors and APIs, learn how to insert objects, and discover the scenarios in which each container excels.
- **Module 3: Non-Modifying STL Algorithms** – Get acquainted with non-modifying algorithms such as `for_each`, `find`, `count`, and `search`. These algorithms allow for iterating through containers without altering their contents, emphasizing compatibility across different container types.
- **Module 4: Modifying STL Algorithms** – Delve into algorithms that modify container contents, including `transform`, `copy`, `remove`, and `unique`. Understand how these algorithms can manipulate data within containers, with a focus on practical examples and container compatibility.

- **Module 5: Sorting STL Operations** – Learn about STL's sorting capabilities through algorithms like `sort`, `stable_sort`, and `binary_search`. This module covers sorting objects and ensuring compatibility across various containers.
- **Module 6: STL Merge Operations** – Focus on merging algorithms such as `merge` and `inplace_merge` alongside operations for sets. Discover how to combine, compare, and retrieve data from different containers effectively.
- **Module 7: STL Utilities and Functional Library** – The STL is more than just containers and algorithms; it also includes a suite of "small" tools and useful functors. This module introduces these utilities, offering practical applications and examples.
- **Module 8: STL Advanced I/O** – Advanced input and output operations play a crucial role in C++ programming. Learn about the classes facilitating I/O, including console and file I/O, string processing, and how to format output for various needs.
- **Module 9: Templates** – Templates are a powerful feature of C++ allowing for type-independent code. Understand the basics of templates, including their syntax, and how to create and use function and class templates. Address common issues and scenarios where templates prove invaluable.

## Course Objectives

### Module 1: STL Sequential Containers

- **Remember** and **understand** the types of sequential containers (`vector`, `deque`, `list`) and their APIs.
- **Apply** knowledge of sequential container adapters (`stack`, `queue`, `priority queue`) in practical scenarios.
- **Analyze** the suitability of using different sequential containers for various data management needs.

### Module 2: STL Associative Containers

- **Understand** the characteristics and behaviors of associative containers (`set`, `multiset`, `map`, `multimap`).
- **Evaluate** the API functionalities of associative containers for efficient data manipulation and retrieval.
- **Create** solutions that effectively implement associative containers for complex data types.

### Module 3: Non-Modifying STL Algorithms

- **Remember** the list of non-modifying algorithms (`for_each`, `find`, `count`, etc.).
- **Apply** non-modifying algorithms to iterate through containers without altering their contents.
- **Analyze** the compatibility and efficiency of non-modifying algorithms across different STL containers.

### Module 4: Modifying STL Algorithms

- **Understand** the purpose and functionality of modifying algorithms (`transform`, `copy`, `remove`, etc.).
- **Apply** these algorithms to manipulate and modify data within containers.

- **Evaluate** scenarios for their appropriate use and understand the implications on container states.

### Module 5: Sorting STL Operations

- **Comprehend** the sorting algorithms available in STL (sort, stable\_sort, binary\_search).
- **Apply** sorting algorithms to organize data within containers.
- **Analyze** and **judge** the best sorting algorithm to use based on the specific requirements of data integrity and order.

### Module 6: STL Merge Operations

- **Understand** the functionality of merging algorithms (merge, inplace\_merge) and set operations.
- **Design** and **implement** strategies to merge datasets effectively using STL operations.
- **Evaluate** the efficiency of merge operations in different use cases.

### Module 7: STL Utilities and Functional Library

- **Identify** and **describe** the utility tools and functors provided by STL for data transformation.
- **Use** these utilities in appropriate contexts to enhance the efficiency and readability of C++ programs.

### Module 8: STL Advanced I/O

- **Understand** advanced I/O classes and operations within STL for handling input and output.
- **Apply** formatting and manipulation techniques to manage console and file I/O operations.
- **Create** I/O operations that are optimized for different data processing requirements.

### Module 9: Templates

- **Comprehend** the concept of C++ templates, including their syntax and use cases.
- **Apply** and **construct** function and class templates to solve generic programming problems.
- **Analyze** and **adapt** template use to create efficient, reusable code for various data types.

## Equipment Requirements

- **Computer with Internet Connection:** Essential for accessing online course materials and participating in course activities.
- **Modern Web Browser:** Required for optimal interaction with online course content.
- **Integrated Development Environment (IDE):** For certain labs and concepts, a local IDE setup is necessary. Although the course is accessible online, engaging with specific programming tasks mandates the use of an IDE installed on your local machine. Detailed guidance on selecting and setting up a suitable IDE, tailored to the advanced concepts covered, will be provided at the beginning of the course.

## Course Outline

MODULE NUMBER AND NAME	RESOURCES	OBJECTIVES COVERED
Welcome to C++ Advanced	Course Structure, Certification Opportunities, and Getting Started with C++ Advanced	Discover the structure of the C++ Advanced course, the journey to CPP certification, and the essential tools for navigating through advanced C++ topics. Learn to utilize the resources provided to build on the knowledge from intermediate C++ courses.
Module 1: STL Sequential Containers	Study Pages, Labs, and Module 1 Test	Understand and apply STL sequential containers like vector, deque, and list along with their APIs. Learn the use of container adapters such as stack, queue, and priority queue, handling objects within these containers, and choosing the right container for specific tasks.
Module 2: STL Associative Containers	Study Pages, Labs, and Module 2 Test	Dive into associative containers including set, multiset, map, and multimap. Explore their behaviors, APIs, and best practices for inserting and managing objects. Determine the appropriate scenarios for using each type of container.
Module 3: Non-Modifying STL Algorithms	Study Pages, Labs, and Module 3 Test	Master non-modifying algorithms for iterating and analyzing container data without alteration. Focus on algorithms like for_each, find, and count, understanding their application across various containers.
Module 4: Modifying STL Algorithms	Study Pages, Labs, and Module 4 Test	Explore modifying algorithms that alter container contents, such as transform, copy, and remove. Assess their impact on data manipulation and container state,

		learning when and how to effectively implement them.
Module 5: Sorting STL Operations	Study Pages, Labs, and Module 5 Test	Grasp sorting mechanisms within STL through algorithms like <code>sort</code> and <code>stable_sort</code> . Understand how to perform binary searches with <code>lower_bound</code> and <code>upper_bound</code> , ensuring data is properly ordered for efficient access.
Module 6: STL Merge Operations	Study Pages, Labs, and Module 6 Test	Learn merging techniques with algorithms such as <code>merge</code> and <code>inplace_merge</code> , and manage sets with operations like <code>includes</code> and <code>min_element</code> . Apply these to combine and compare collections efficiently.
Module 7: STL Utilities and Functional Library	Study Pages, Labs, and Module 7 Test	Utilize STL's utility tools and functors for data transformation and manipulation. Apply practical examples to see these utilities in action, enhancing program functionality and efficiency.
Module 8: STL Advanced I/O	Study Pages, Labs, and Module 8 Test	Advance your input and output operations using STL. Cover topics including stream manipulation, file I/O, and string processing, aiming to improve data handling and presentation in C++ applications.
Module 9: Templates	Study Pages, Labs, and Module 9 Test	Delve into C++ templates, learning to define and use function and class templates for generic programming. Address typical challenges and evaluate scenarios for their effective use.
Course Completion	C++ Advanced – Final Test (Score 70% or more to qualify for a 50%	By completing the C++ Advanced final test successfully, you demonstrate a deep understanding and application of

	discount on the CPP exam) + Become <a href="#">CPP</a> certified (Paid Option)	complex C++ programming concepts. This completion sets the stage for success in the CPP certification exam and prepares you for tackling sophisticated software development projects.
--	--	---

## Why C++ Advanced?

Advancing your knowledge through the *C++ Advanced* course not only broadens your programming skills but also prepares you for high-level challenges in various tech domains. This deeper understanding of C++ enhances your ability to develop sophisticated software, contributing to your growth as a software developer, system architect, or in other advanced roles.

Achieving the CPP certification as a result of this course distinguishes you as a proficient C++ programmer, ready to tackle complex projects and stand out in the tech industry.