CPP: Advanced Programming in C++

Scope and Sequence

Last updated September 4, 2017

Scope and Sequence - Table of Contents

Target Audience

The *CPP: Advanced Programming in C++* curriculum is designed for students who already have a good knowledge of the C++ language, including inheritance and operator overloading, and want to learn the more advanced C++ topics such as templates and the Standard Template Library.

Prerequisites

There are no formal prerequisites for this course. However, it is recommended that the student complete the *CPA: Programming Essentials in C++* course prior to signing up for the *CPP: Advanced Programming in C++* course.

Target Certification

The *CPP: Advanced Programming in C++* curriculum helps students prepare for the *CPP – C++ Certified Professional Programmer* certification exam. C++ Certified Professional Programmer (CPP) is a professional certification that measures the student's ability to accomplish coding tasks related to the more advanced C++ topics, i.e., templates and the Standard Template Library.

Curriculum Description

The *CPP: Advanced Programming in C++* course familiarizes students with advanced programming techniques, customs, and vocabulary as well as advanced library functions.

The course is broken down into 9 modules:

- Module 1: STL Sequential Containers
- Module 2: Associative STL Containers
- Module 3: Non-modifying STL Operations
- Module 4: Modifying STL Algorithms (Operations)
- Module 5: Sorting STL Algorithms
- Module 6: Merging STL Algorithms
- Module 7: Utilities and Functional Tools in the STL
- Module 8: Advanced Input and Output
- Module 9: Templates

Each student has access to hands-on practice materials, quizzes, and assessments to learn how to utilize the skills and knowledge gained on the course and interact with some real-life programming tasks and situations.

Curriculum Objectives

The aim of the course is to familiarize the student with the C++ template mechanism, reading and understanding definitions of template functions and classes, using property template classes and methods including third party templates, creating template functions and classes, the C++ STL library including the I/O part, and solving common programming problems with predefined STL classes and methods.

After completing this course, the student should be able to:

- create a vector, deque and list;
- fill a vector, deque, and list with values;
- remove values from collections (vector, deque and list);
- use collections (vector, deque and list) in standard ways (e.g., iterate them);
- create a map, set, multimap and multiset;
- fill a map, set, multimap, and multiset with values;

- remove some values from collections (map, set, multimap and multiset);
- use collections (map, set, multimap and multiset) in standard ways (e.g., find an element);
- prepare data collections for data processing;
- use search_n, mismatch, count_if, and find operations for data processing;
- use for_each to iterate over data collections;
- use copy, unique_copy and reverse_copy operations for data copying;
- use generate and generate_n operations for data creation;
- use iter_swap, swap_ranges, rotate, random_shuffle, and transform operations for data processing;
- use replace and replace_if operations for data replacement;
- use binary_search operations for quick searches for data;
- use sort and stable_sort operations for data sorting;
- use lower_bound and upper_bound for finding boundaries;
- use equal_range for finding equal ranges;
- use set-related operations for finding the maximum and minimum of sets;
- use include operations to compare two collections (e.g., vector);
- use merge and inplace_merge operations to merge data from collections;
- use bind2nd and ptr_fun to process data;
- use comparison operators and functional operations to transform data
- format cout output;
- use cout flags and manipulators;
- use stringstream and getline to process string data;
- read from and write to files with specific flags;
- create a function template and use it;
- create a class template and use it;
- create a class template with specialized methods;
- create a class template and a specialized function which uses this template.

Course Outline

Learning Module	CPP – C++ Certified Professional Programmer Certification Objectives Covered
1 – STL Sequential containers	• Types of sequential containers, • vector, deque, list and their APIs, • Sequential container adapters – stack, queue and priority queue, • Dealing with objects as container elements, • Usage – when to use what.
2 – STL Associative containers	• Types of associative containers, • set and multiset – behavior and API, • map and multimap – behavior and API, • Putting objects into set and map, • Usage – when to use what.
3 – Non-modifying STL algorithms	• Definition of a non-modifying algorithm,

	• List of non-modifying algorithms: for_each, find, find_if, find_end, find_first_of, adjacent_find, count, count_if, mismatch, equal, search, search_n, • Examples, • Container compatibility.
4 – Modifying STL algorithms	• Definition of a modifying algorithm, • List of modifying algorithms: transform, copy, copy_backward, swap, swap_ranges, iter_swap, replace, fill, fill_n, generate, generate_n, remove, remove_if, unique, unique_copy, reverse, reverse_copy, rotate, partition, stable_partition, • Examples, • Container compatibility.
5 – Sorting STL operations	• List of sorting algorithms: random_shuffle, sort, stable_partition, lower_bound, upper_bound, equal_range, binary_search, • Examples, • Containers compatibility, • Sorting of objects.
6 – STL merge operations	• List of merging algorithms: merge, includes, min_element, max_element, inplace_merge, • STL operations for sets, • Examples, • Container compatibility.
7 – STL utilities and functional library	• STL “small” tools, • List of useful functors, • Examples.
8 – STL advanced I/O	• Classes which provide input and output capability, • Console I/O, • Formatting, • File I/O, • Strings I/O, • Examples.
9 – Templates	• What are templates? • Basic syntax, • Function templates, • Class templates, • When to use templates, • Typical problems when using templates.

Minimum System Requirements

The course content modules, labs, quizzes and assessments can be accessed online through any Internet browser. For the best learning experience, we recommend using the most recent versions of Mozilla Firefox or Internet Explorer/Microsoft Edge.

Industry certification

The course curriculum helps students prepare for the C++ Institute *CPP – C++ Certified Professional Programmer* certification.

A Statement of Achievement will be issued to participants who successfully complete the *CPP: Advanced Programming in C++* course. The Statement of Achievement will acknowledge that the individual has completed the course and is now ready to attempt the qualification *CPP – C++ Certified Professional Programmer Certification*, taken through Pearson VUE computer-based testing, at a 51% discount.

To receive the Statement of Achievement, instructors must mark the student as having successfully passed the course.

For additional information about the *C++ Institute CPP – C++ Certified Professional Programmer certification*, please visit www.cppinstitute.org/certification.