

# C++ - Advanced Programming with STL

---

**Length:**

5 days

**Description:**

With this course students will learn the advanced features of the C++ standard template library (STL). The course covers techniques for extending the STL and the C++ language. Students will learn advanced STL features including the implementation of iterator and function object adapters. Advanced template programming techniques used in the STL design will be covered as well.

**Course Objective:**

By taking this course student will:

- Understand the concepts and architecture of the STL
- Learn how to take advantage of the most commonly useful parts of the STL
- Know how to avoid falling into common STL traps
- Learn compiler diagnostics, debugging and workarounds for environment-specific limitations
- Study containers and the differences among them
- Learn how to use iterators, function objects and adapters
- Study STL algorithms and common STL extensions

**Audience:**

Software engineers, designers and programmers who have experience programming in C++.

**Prerequisites:**

Experience with the C++ programming language. Introduction to STL and relational database are helpful.

**Course Contents****Chapter 1 - Templates**

- Function template
- Class template

**Chapter 2 - How to generalize a function**

- The find algorithm, version 1
- Assumptions, version 1
- Transformation
- The find algorithm, version 2
- Assumptions, version 2
- Accessing builtin array elements
- Transformation
- The find algorithm, version 3
- Assumptions, version 3
- Transformation
- The find algorithm, version 4
- Assumptions, version 4
- Transformation
- The find algorithm, version 5
- Assumptions, version 5
- Replacing builtin pointers with iterators
- Transformation
- The find algorithm, version 6
- Assumptions, version 6

- Container
- Transformation
- The find algorithm, version 7
- Assumptions, version 7

### **Chapter 3 - Functors**

- How to create an object that behaves like a function

### **Chapter 4 - STL - The basics**

- What is STL?
- Kinds of containers
- vector's implementation
- vector
- The container vector
- Notes
- list
- The container list
- Notes
- deque
- The container deque
- set
- The container set
- multiset
- maps and pairs
- map
- The container map
- multimap
- Differences among containers
- How to make your own algorithm

### **Chapter 5 - Smart pointers**

- Smart pointer
- Scoped pointer
- Auto pointer

### **Chapter 6 - Concepts and Architecture of the STL**

- Pointers into arrays
- Searching an array
- Using find
- Generalizing find
- Generalizing find again
- std::find
- STL algorithms
- find and containers
- Using find with list
- find and other containers
- STL and conventions
- The extensibility of the STL
- Generalizing find further
- Using find\_if
- Function objects
- What is "The STL"?
- STL headers
- STL and std
- A sample program

- The same program with using declarations
- The same program with a using directive
- Headers, std, and course examples
- const\_iterators
- Sample program with const\_iterator
- iterators, const\_iterators, and algorithms
- Thread-safety and the STL

## **Chapter 7 - Containers I - Introduction and Sequence Containers**

- Container adapters
- Standard containers
- Universal standard container functionality
- Summary of universal standard container functionality
- The significance of vector and string
- Size versus capacity
- Performing "shrink to fit" on vectors and strings
- vector
- vector operations
- vector insertion
- Summary of vector operations
- vector and array-based functions
- vector<bool>
- string
- string and array-based functions
- A (silly) string example
- string functionality and the STL
- deque
- deque vs. vector
- list
- list and iterator/pointer/reference invalidation
- Splicing lists
- List specializations of general algorithms

## **Chapter 8 - Containers II - Associative Containers**

- Associative containers
- Comparison functions
- Specifying comparison functions
- Comparison functions and containers of pointers
- Equality vs. equivalence
- The utility class pair
- Set and multiset search functions
- Set and multiset insert functions
- Set and multiset erase functions
- Maps and multimaps
- Maps, multimaps, sets, and multisets
- Map and multimap functionality
- Operator[] for maps
- The constness of associative container elements
- An alternative to standard associative containers
- General container functionality rules

## **Chapter 9 - Iterators**

- Beyond iterators and const\_iterators
- Reverse iterators
- Converting reverse\_iterators to iterators

- The iterator/reverse\_iterator offset
- Converting iterators to const\_iterators
- Converting const\_iterators to iterators
- Stream iterators
- istream\_iterators
- C++'s most vexing parse
- ostream\_iterators
- An STL approach to the UNIX cat command
- An STL approximation of UNIX's sort | uniq
- Insert iterators
- back\_inserter
- front\_inserter
- inserter
- Iterator categories
- Which iterators are in which categories
- Iterators vs. pointers

### **Chapter 10 - Function objects and adapters**

- Function object adapters
- Supporting function object adapters
- Making built-in functions adaptable
- Passing nonstatic member functions to algorithms
- Passing member functions to algorithms
- Binders, mem\_fun, and mem\_fun\_ref
- Standard function objects

### **Chapter 11 - Algorithms**

- The scope of STL algorithms
- Algorithms we've already seen
- Member functions vs. non-member functions
- Game plan
- for\_each
- transform
- remove
- Searching algorithms
- Reordering algorithms
- Modifying algorithms
- Removing algorithms
- Copying algorithms
- Creation algorithms
- Summarizing algorithms

### **Chapter 12 - Common STL extensions**

- STL extensions
- Nonstandard hashed containers
- The non-standard hashed containers
- SGI's hashed containers: Interface
- Dinkumware's hashed containers: Interface
- Declaring simple hashed containers
- SGI's hashed containers: Implementation
- Dinkumware's hashed containers: Implementation
- Hashed container summary
- Other nonstandard STL containers
- Common nonstandard function object adapters

## **Chapter 13 - STL information sources**

- Books
- Implementation articles by P.J. Plauger
- Other articles
- Web sites
- Newsgroups
- The last words on everything C++