Publication date : 04/17/2024

Course : C++ programming, advanced

Practical course - 4d - 28h00 - Ref. POP
Price : 2240 € E.T.

4,8 / 5

BEST

Constantly evolving, from C++98 to C++20, the C++ language offers mechanisms for robust and rich design. Recent C++ standards significantly enhance the Standard Template Library (STL). This training course will enable you to deepen your understanding of C++ design by learning about the latest developments in the language and making effective use of the STL.

Teaching objectives

At the end of the training, the participant will be able to:

- Find out what's new in each version
- Managing memory, pointers and references
- Implementing genericity in C++
- Discover the STL standard library
- Use the contributions of the C++11 standard

Intended audience

C++ application designers and developers, project managers, software architects.

Prerequisites

Good knowledge of C++ development, or knowledge equivalent to that acquired in the course "Object Programming in C++" (ref. C++). Experience required.

Practical details

Hands-on work

The course will take place on Windows/Visual C++ workstations. Numerous exercises will enable you to put into practice the topics covered, more specifically from a design point of view.

Course schedule

PARTICIPANTS

C++ application designers and developers, project managers, software architects.

PREREQUISITES

Good knowledge of C++ development, or knowledge equivalent to that acquired in the course "Object Programming in C++" (ref. C++). Experience required.

TRAINER QUALIFICATIONS

The experts leading the training are specialists in the covered subjects. They have been approved by our instructional teams for both their professional knowledge and their teaching ability, for each course they teach. They have at least five to ten years of experience in their field and hold (or have held) decision-making positions in companies.

ASSESSMENT TERMS

The trainer evaluates each participant's academic progress throughout the training using multiple choice, scenarios, hands-on work and more.
Participants also complete a placement test before and after the course to measure the skills they've developed.

1 Reminders

- Memory allocation classes.
- Object construction, initialization and loading.
- Memory leaks.
- Constance, the mutable keyword, Lazy Computation.
- Friendship C++ and access control.
- Virtual destruction.
- Exception management strategy.
- Namespaces.

2 New language features in C++11

- nullptr and other literals.
- The =delete, =default directives.
- Manufacturer delegation.
- Enumerations "type safe".
- The auto keyword and loop over an interval.
- Reference rvalue and impact on the normal form of C++ classes.
- Lambda expressions.

Hands-on work

Rewriting existing C++ code in C++11, comparing the two implementations.

3 Operator management

- Binary and unary operators.
- The indirection operator, use case.
- The referencing operator.
- Prefixed and postfixed increment/decrement operators.
- Other operators: comparison, assignment...
- Overloading of the [] operator, insertion operators (<<) and extraction operators (>>).
- Functors and operator overloading (), an advantage over functions.

Hands-on work

Creation of functors and proxies (memory release, reference counting) with the operators studied.

4 Conversion and RTTI

- Conversion operators. Implicit constructions, the explicit keyword.
- Casting operators const_cast, static_cast, reinterpret_cast.
- Dynamic conversion and Runtime Type Information.
- The typeid operator and related exceptions.
- The type_info class.
- Control of "downcasting" using the dynamic_cast operator.

Hands-on work

Implementation of the idioms "is-a" and "is-kind-of" with dynamic_cast.

TEACHING AIDS AND TECHNICAL RESOURCES

• The main teaching aids and instructional methods used in the training are audiovisual aids, documentation and course material, hands-on application exercises and corrected exercises for practical training courses, case studies and coverage of real cases for training seminars.

• At the end of each course or seminar, ORSYS provides participants with a course evaluation questionnaire that is analysed by our instructional teams.

• A check-in sheet for each half-day of attendance is provided at the end of the training, along with a course completion certificate if the trainee attended the entire session.

TERMS AND DEADLINES

Registration must be completed 24 hours before the start of the training.

ACCESSIBILITY FOR PEOPLE WITH DISABILITIES

Do you need special accessibility accommodations? Contact Mrs. Fosse, Disability Manager, at psh-accueil@orsys.fr to review your request and its feasibility.

5 Genericity

- Introduction to class patterns. Genericity and preprocessors.
- Generic function. Generic class. Generic composition. Generic generalization.
- Partial and total specialization.
- Introduction to meta-programming.
- Genericity, the unifying principle of the STL and Boost libraries.

Hands-on work

Start of case study to be completed with STL. Implementation of generic composition and generalization. Creation of generic plug-ins.

6 STL (Standard Template Library)

- STL components: complementary types, containers, algorithms, iterators, function objects, adapters.
- STL strings, the basic_string template class and its specializations.
- Sequential and associative containers: definition, role and selection criteria.
- Allocators and container memory management.
- Methods for inserting, deleting, iterating and accessing the main containers: Vector, List, Set, Stack...
- The iterator concept. Container traversal.
- The different groups of STL algorithms: non-mutant, mutant, sorting and merging, numerical.
- Container handling (manipulation, value searches, etc.).
- Parameterize generic algorithms with "function" objects.
- Adapters" and modifying a component's behavior.
- STL and stream processing (files, memory, etc.).
- RAII principle: automatic pointers and the auto_ptr class.
- Standard STL exceptions.

Hands-on work

Implement relationships with STL collections. Use of any standard algorithms.

7 New C++11 features in the standard library

- Historical trends: Boost --> TR1 --> C++11.
- New containers: array, forward_list, unordered_set, unordered_map.
- The tuple class.
- Smart pointers: shared_ptr, weak_ptr, unique_ptr.
- New functors and binders.
- Introduction to thread management.
- Regular expressions.

Hands-on work

Implementing robustness with smart pointers. Using regular expressions.

8 Boost and its principles

- The Pointer Container Library (destruction of container pointer data).
- The boost::any and boost::variant data structures.
- Event-driven programming (connections and signals).
- Process management, inter-process communication mechanisms and shared memory.

Hands-on work
Improved case study implementation using the Pointer Container Library.

9 Advanced use of inheritance

- Inheritance versus boarding. Private heritage. Protected heritage.
- Exporting hidden members with the Using Clause.
- Multiple inheritance and member collision management.
- Diamond inheritance. Virtual inheritance and dynamic_cast.
- Design principles: Liskov substitution, open/close principle, dependency inversion.
- Interface implementation rules in C++.

Hands-on work
Combine multiple, private and export inheritance to design robust, highly scalable classes.

Dates and locations

REMOTE CLASS
2026 : 26 May, 13 Oct., 17 Nov.

PARIS LA DÉFENSE
2026 : 26 May, 13 Oct., 17 Nov.