+33 (0)1 48 10 09 46
formation@ambient-it.net
140 Avenue Jean Lolive 93500 Pantin - France

Updated 06/06/2025

Sign up

C++ Object programming training

5 days (35 hours)

Presentation

Our training course in C++ object-oriented programming will enable you to master the fundamental principles of object-oriented development while discovering the advanced features of the C++ language. Every day, you'll learn how to structure your code in a clear, high-performance and scalable way.

Discover our Object-Oriented Programming course in C++, designed to help you gradually master the essential fundamentals of the language. You'll progress step by step, from the basics to advanced concepts, while constantly reinforcing your skills.

Start with a clear immersion in fundamental C++ syntax and recent standards, learning all the subtleties of object-oriented programming as well as the best practices of factoring in the SOLID and Clean Code principles.

As the course progresses, you'll master the control structures, data types and basic to advanced memory management essential for writing robust, high-performance programs.

Apply the principles of Object-Oriented Design: classes, encapsulation, inheritance and polymorphism. Each practical workshop will enable you to progressively develop your technical skills in this modern language. You'll also discover its rich ecosystem through popular libraries such as Boost and Poco.

Finally, by progressively integrating modern C++ innovations (C++17 to C++23 standards), you'll acquire skills that are immediately applicable in a professional environment, ensuring your ability to design applications that will last over time.

Our program covers all the essential notions of C++, from conceptual basics and fundamental syntax to advanced practices based on modern language standards. You'll be able to design applications

memory management and design patterns.

Like all our training courses, this one includes lots of practical exercises, and will introduce you to the latest version of C++, C++ 23.

Objectives

- Understand C++ syntax and fundamental concepts
- Master the major additions to the C++ standards
- Apply the principles of Object-Oriented Design
- Write simple programs using good development practices
- Using control structures and data types in C++
- Basic file and memory handling

Target audience

- Developers,
- Engineers,
- Development project managers

Prerequisites

- Knowledge of object-oriented programming principles and C++ experience

C++ Object training program

Day 1: In-depth introduction and essential syntax

Discover modern C++

- History and recent developments
- C++ language overview
- Environment setup and configuration
- Ecosystem (IDE, package managers)
- Reminder of best development practices (Factoring, Clean Code, SOLID, KISS, etc.).
- Recent ISO standards
- Practical workshop: Initializing the ProBank project with basic compilation.

Major standards

- C++98/03: First official standardization. Initial STL.
- C++11: Smart pointers, lambdas, auto, constexpr, multithreading.
- C++14: constexpr improvements, improved auto, generic lambda functions.

- C++17: optional, variant, any, Structured bindings, standard filesystem, parallel algorithms.
- C++20: Modules, coroutines, concepts. Ranges and advanced views.
- C++23: std::print, std::expected, changes to modules, new views (views::zip, etc.).
- C++26: Next standards expected (Reflexion, Contracts, Pack indexing, advanced constexpr, improved internationalization)

Basic and advanced types

- Primitive and derived types
- Range and service life
- Const and constexpr
- Auto and type deduction
- Enum class and strong types
- Common operators
- Practical workshop: Definition and management of initial customer data.

Control structures

- Advanced conditions (if constexpr, modern switch)
- Classic and modern buckles
- Range-based loops (C++11+)
- Flow management (break, continuous)
- Basic pattern matching
- Practical workshop: Implementing robust data entry validation.

OOP basics in C++: Classes and Objects

- Creating and managing classes and objects
- Encapsulation
- Builders, destroyers and builder overload
- Notions of single and multiple inheritance
- Polymorphism of virtual functions
- Static methods
- Static limbs
- Case study: First steps in structuring responsibilities and data for a BankAccount Day 2:

Modularity and expert memory management

Advanced functions

- Namespaces
- Classic and inline functions
- Passage by reference vs. value
- Default values
- Function overload
- Recursive functions
- Lambda expressions and capture
- Practical workshop: Advanced modularization of banking functions.

Advanced memory management

- Dynamic allocation
- Pointers and smart pointers
- Stack vs Heap
- References and const correctness
- Resource Acquisition Is Initialization
- RAII and automatic management
- Memory leak prevention and detection
- Practical workshop: "Memory leak hunter": correcting a complex program that deliberately contains hidden memory errors

I/O and advanced file management

- Reading and writing text files
- File streams and buffers
- IO error handling
- Basic serialization of objects in files
- Basic CSV/XML file processing
- Disk access optimization and performance best practices
- Practical workshop: Optimized backup and restoration of customer data.

Day 3: Deepening Object-Oriented Programming

Classes and advanced encapsulation

- Declaration and strict encapsulation
- Builders and manufacturers
- Static members and const methods
- Mutators and accessors
- Pattern builder
- Practical workshop: Advanced structuring of the CompteBancaire class.

Advanced inheritance and dynamic polymorphism

- Simple multiple inheritance
- Polymorphism and abstract classes
- Virtual and override methods
- Interface management (abstract class)
- Dynamic_cast and polymorphic management
- Practical workshop: Hierarchical creation of current and savings accounts.

C++ design pattern

- Maintenance and upgradability benefits
- The 3 main families of Design Patterns
  - Designer
  - Structural
  - Behavioral
- The main Patterns useful in software development

- Factory design patterns, Singleton
- Structuring patterns Adapter, Decorator
- Behavior patterns Observer, Strategy
- Good design practices
- Practical workshop: Using Singleton and Factory patterns to manage bank accounts.

Day 4: Advanced techniques, STL and popular libraries

Robust exception handling

- Standard exception
- Customizing exceptions
- Good capture practices
- Automated clean-up
- Customized business errors
- Practical workshop: Advanced exception handling on transactions.

In-depth generic programming

- Templates functions/classes
- Specializations and optimizations
- Concepts (C++20)
- Constraints and performance
- Software development benefits
- Workshop: Generic template for managing multiple transactions.

STL: Mastering containers and advanced algorithms

- Vectors, Lists, Maps, Advanced Sets
- Complex algorithms (sort, transform)
- Iterators and ranges (C++20)
- Performance and choice of containers
- Lambdas and STL
- Practical workshop: Efficient implementation of transaction history management.

STL: Mastering containers and advanced algorithms

- Boost (filesystem, algorithm)
- Poco (network, security)
- Practical benefits
- Fast integration
- Typical use cases
- Practical workshop: Using Boost.Filesystem to manage bank files

. Day 5: Modernity in C++ (C++17 to C++23), Cross-platform and Internationalization

Modern C++17 techniques

- Structured Bindings
- Optional, variant, any
- Filesystem standard
- Parallel algorithms
- constexpr functions
- Practical workshop: Advanced modernization of existing ProBank code.

Innovations in C++20 and C++23

- Modules and partitioning
- Asynchronous coroutines
- Ranges and advanced views
- What's new in C++23 (std::print, expected)
- Practical workshop: Experimental integration of modules and coroutines in ProBank.

Cross-platform management and internationalization

- Cross-platform compilation (Windows, Linux, macOS)
- Managing external dependencies
- Internationalization (gettext, ICU)
- Encodings and UTF-8
- Managing system differences
- Practical workshop: Adapting ProBank for international, multi-platform use.

Further information

Companies concerned

This course is aimed at both individuals and companies, large or small, wishing to train their teams in a new advanced computer technology, or to acquire specific business knowledge or modern methods.

Positioning on entry to training

Positioning at the start of training complies with Qualiopi quality criteria. As soon as registration is finalized, the learner receives a self-assessment questionnaire which enables us to assess his or her estimated level of proficiency in different types of technology, as well as his or her expectations and personal objectives for the training to come, within the limits imposed by the selected format. This questionnaire also enables us to anticipate any connection or security difficulties within the company (intra-company or virtual classroom) which could be problematic for the follow-up and smooth running of the training session.

Teaching methods

Practical course: 60% Practical, 40% Theory. Training material distributed in digital format to all participants.

Organization

The course alternates theoretical input from the trainer, supported by examples, with brainstorming sessions and group work.

Validation

At the end of the session, a multiple-choice questionnaire verifies the correct acquisition of skills.

Sanction

A certificate will be issued to each trainee who completes the course.