Course : C++, Object programming

Practical course - 5d - 35h00 - Ref. CGE
Price : 3070 CHF E.T.

NEW

On completion of the course, participants will be able to apply the fundamental principles of object-oriented design and design applications in C++. This training program is intended for employees of professional branches covered by the OPCO Atlas.

Teaching objectives

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

- Understand the syntax and fundamental concepts of C++
- 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

Intended audience

For OPCO Atlas members: developers, engineers, project managers with close links to development.

Prerequisites

Familiarity with the principles of object-oriented programming (OOP) and experience of a programming language...

Practical details

Hands-on work

Case studies and practical exercises.

Teaching methods

70% pratique – 30% théorie. Pour optimiser le parcours d'apprentissage, des modules e-learning peuvent être fournis avant et après la session présentielle ou la classe virtuelle, sur simple demande du participant.

Course schedule

PARTICIPANTS

For OPCO Atlas members: developers, engineers, project managers with close links to development.

PREREQUISITES

Familiarity with the principles of object-oriented programming (OOP) and experience of a programming language...

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 Java, learning object-oriented programming - Pre-training digital learning content

- Introduction.
- Classes.
- Inheritance.

Digital activities

Object-oriented programming (OOP) is a paradigm now present in all modern programming languages. These concepts enable the production of efficient, powerful and easy-to-maintain code. This online training course introduces the key concepts of object-oriented programming, such as the notion of class and inheritance.

2 A reminder of how C++ works

- Presentation of the C++ language and its evolutions.
- Install tools (compiler, IDE, project manager).
- Structure of a C++ program.
- Compilation, execution, source file management.

Hands-on work

Installation et configuration de l'environnement. Structure et syntaxe de base. Exercices de contrôle de flux.

3 Tables, strings and data management

- Static and dynamic tables.
- Character strings (C and C++).
- Standard I/O and files.

Hands-on work

Manipulation de données. Exercices sur les structures de données. Optimisation et bonnes pratiques.

4 Object-oriented programming

- Classes and objects.
- Encapsulation, abstraction.
- Builders, destroyers.
- Static and instance members.

Hands-on work

Création de classes simples. Héritage et polymorphisme. Cas d'usage avancés.

5 Testing and optimization

- Unit testing with C++ frameworks (Catch2, GoogleTest).
- Performance optimization techniques.
- Memory management strategies.

Hands-on work

Tests unitaires avancés. Optimisation des performances. Déploiement optimisé.

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.

6 Memory management in C++

- Dynamic allocation (new, delete).
- Pointers, references, smart pointers.
- Memory leaks, resource management (RAII).
- Good memory management practices.

Hands-on work
Manipulation de la mémoire. Atelier RAII et gestion des ressources. Optimisation mémoire.

7 Introduction to STL (Standard Template Library)

- Introducing the STL and its benefits.
- Main containers: vector, list, map, set.
- Iterators and collection paths.
- Standard algorithms (sort, find, etc.).

Hands-on work
Manipulation des conteneurs STL. Exercices sur les algorithmes STL. Optimisation et bonnes pratiques STL.

8 Advanced patterns and design

- Classic design patterns (Singleton, Factory, Observer, etc.).
- Using templates for generic patterns.
- Best practices in object-oriented design.

Hands-on work
Implémentation de patterns. Patterns avancés et templates. Cas d'usage et revue de code.

9 Advanced testing and optimization

- Advanced unit testing (mocks, parameterized tests).
- Performance optimization (profiling, code analysis).
- Refactoring strategies.

Hands-on work
Tests avancés. Optimisation et refactoring. Déploiement et synthèse.

10 Generic programming and templates

- Function and class templates.
- Specialization and template overload.
- Variadic templates and basic metaprogramming.
- Best practices and pitfalls to avoid.

Hands-on work
Création de templates. Métaprogrammation et templates avancés. Optimisation et bonnes pratiques.

11 Exception handling and robustness

- Exception handling (try, catch, throw).
- Customized exceptions.
- Best practices in error management.
- Impact on performance and legibility.

Hands-on work
Manipulation des exceptions. Exercices sur la robustesse. Bonnes pratiques et revue de code.

12 Integration of complex projects

- Organization of a multi-file project.
- Use of CMake or other build tools.
- Dependency management and modularity.
- Automated documentation and testing.

Hands-on work
Structuration d'un projet. Intégration et gestion des dépendances. Cas d'usage et revue de projet.

13 Testing, CI/CD and synthesis

- Advanced unit and integration testing.
- Introduction to continuous integration (CI/CD).
- Automated build and test tools.
- Synthesis of the day's findings.

Hands-on work
Mise en place de tests automatisés. CI/CD et automatisation. Synthèse et plan d'action.

14 Advanced programming in C++

- Lambda expressions, auto, nullptr, move semantics.
- For-range loops, uniform initialization.
- Advanced smart pointers, resource management.
- Anonymous functions and closures.

Hands-on work
Exercices sur les nouveautés du langage. Ateliers sur la modernisation du code. Optimisation avancée.

15 Safety and robustness in C++

- Memory security (buffer overflow, use-after-free).
- Best practices for input validation.
- Concurrent access management (mutex, threads).
- Security analysis tools.

Hands-on work
Analyse de vulnérabilités. Exercices sur la concurrence. Bonnes pratiques et revue de code.

16 Performance and multithreading

- Introduction to multithreading in C++.
- Use of threads, futures, promises.
- Synchronization and management of shared resources.
- Profiling and performance analysis tools.

Hands-on work
Mise en œuvre du multithreading. Optimisation de la concurrence. Cas d'usage et revue de code.

17 Testing, monitoring and synthesis

- Performance and load testing.
- Monitoring tools (Valgrind, perf, etc.).
- Log analysis and anomaly detection.
- Synthesis of the day's findings.

Hands-on work
Tests de performance. Monitoring et analyse. Synthèse et plan d'action.

18 Summary project

- Analysis of specifications.
- Modular, object-oriented design.
- Development of a complete C++ application.
- Test integration, optimization and documentation.

Hands-on work
Réalisation du projet. Soutenance et retours.

19 Consolidating best practices

- Good development practices C++.
- Error and exception handling.
- Technical and user documentation.
- Maintenance and evolution planning.

Hands-on work
Revue de code croisée. Atelier documentation et maintenance. Synthèse et bonnes pratiques.

20 Personal action plan and closing

- Setting personal goals.
- Identify resources and tools for progress.
- Planning for practical use.
- On-the-spot evaluation and feedback.

Hands-on work
Élaboration du plan d'action personnel. Évaluation et feedback. Clôture et perspectives.

21 UML, learning to model with diagrams - Post-training digital learning content

- Fundamental concepts.
- Structural diagrams.
- Behavioral diagrams.

Digital activities

This online training course presents the fundamentals of object-oriented design, the various UML structural and behavioral diagrams, as well as their objectives and uses. An example of an object-oriented design will be used to put into practice the application of UML to efficiently specify, visualize and document a computer system.

Dates and locations

REMOTE CLASS
2026 : 30 Mar., 22 June, 5 Oct., 14 Dec.