Formation : Campus Atlas - C++, Programmation Objet

Cours pratique - 5j - 35h00 - Réf. CGE
Prix : 2610 € H.T.

NEW

À l'issue de la formation, le participant sera capable de mettre en œuvre les principes fondamentaux de la conception orientée objet et de concevoir des applications en C++. Ce programme de formation est destiné aux salariés des branches professionnelles relevant de l'OPCO Atlas.

Objectifs pédagogiques

À l'issue de la formation, le participant sera en mesure de :

- Comprendre la syntaxe et les concepts fondamentaux du C++
- Maîtriser les ajouts majeurs des normes C++
- Appliquer les principes de la conception orientée objet
- Écrire des programmes simples en appliquant les bonnes pratiques de développement
- Utiliser les structures de contrôle et les types de données en C++
- Manipuler les fichiers et la mémoire de manière basique

Public concerné

Pour les adhérents à l'OPCO Atlas : développeurs, ingénieurs, chefs de projet proches du développement.

Prérequis

Connaître les principes de la programmation orientée objet (POO) et disposer d'une expérience d'un langage de programmation…

Méthodes et moyens pédagogiques

Travaux pratiques

Des études de cas et exercices pratiques.

Méthodes pédagogiques

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.

PARTICIPANTS

Pour les adhérents à l'OPCO Atlas : développeurs, ingénieurs, chefs de projet proches du développement.

PRÉREQUIS

Connaître les principes de la programmation orientée objet (POO) et disposer d'une expérience d'un langage de programmation…

COMPÉTENCES DU FORMATEUR

Les experts qui animent la formation sont des spécialistes des matières abordées. Ils ont été validés par nos équipes pédagogiques tant sur le plan des connaissances métiers que sur celui de la pédagogie, et ce pour chaque cours qu'ils enseignent. Ils ont au minimum cinq à dix années d'expérience dans leur domaine et occupent ou ont occupé des postes à responsabilité en entreprise.

MODALITÉS D'ÉVALUATION

Le formateur évalue la progression pédagogique du participant tout au long de la formation au moyen de QCM, mises en situation, travaux pratiques…
Le participant complète également un test de positionnement en amont et en aval pour valider les compétences acquises.

Modalités d'évaluation

Le formateur évalue la progression pédagogique du participant tout au long de la formation au moyen de QCM, mises en situation, travaux pratiques…

Le participant complète également un test de positionnement en amont et en aval pour valider les compétences acquises.

Programme de la formation

1 Java, apprendre la programmation orientée objet - Contenu digital learning préformation

- Introduction.
- Les classes.
- L'héritage.

Activités digitales

La programmation orientée objet (POO) est un paradigme présent aujourd'hui dans l'ensemble des langages de programmation modernes. Ces concepts permettent de produire un code efficace, puissant et facile à maintenir. Cette formation en ligne présente les concepts clés de la programmation orientée objet comme la notion de classe et d'héritage.

2 Rappel sur le fonctionnement du C++

- Présentation du langage C++ et de ses évolutions.
- Installation des outils (compilateur, IDE, gestionnaire de projet).
- Structure d'un programme C++.
- Compilation, exécution, gestion des fichiers source.

Travaux pratiques

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

3 Tableaux, chaînes et gestion des données

- Tableaux statiques et dynamiques.
- Chaînes de caractères (C et C++).
- Entrées/sorties standard et fichiers.

Travaux pratiques

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

4 Programmation orientée objet

- Classes et objets.
- Encapsulation, abstraction.
- Constructeurs, destructeurs.
- Membres statiques et d'instance.

Travaux pratiques

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

MOYENS PÉDAGOGIQUES ET TECHNIQUES

• Les moyens pédagogiques et les méthodes d'enseignement utilisés sont principalement : aides audiovisuelles, documentation et support de cours, exercices pratiques d'application et corrigés des exercices pour les formations pratiques, études de cas ou présentation de cas réels pour les séminaires de formation.
• À l'issue de chaque formation ou séminaire, ORSYS fournit aux participants un questionnaire d'évaluation du cours qui est ensuite analysé par nos équipes pédagogiques.
• Une feuille d'émargement par demi-journée de présence est fournie en fin de formation ainsi qu'une attestation de fin de formation si le participant a bien assisté à la totalité de la session.

MODALITÉS ET DÉLAIS D'ACCÈS

L'inscription doit être finalisée 24 heures avant le début de la formation.

ACCESSIBILITÉ AUX PERSONNES HANDICAPÉES

Pour toute question ou besoin relatif à l'accessibilité, vous pouvez joindre notre équipe PSH par e-mail à l'adresse psh-accueil@orsys.fr.

5 Testing et optimisation

- Tests unitaires avec frameworks C++ (Catch2, GoogleTest).
- Techniques d'optimisation des performances.
- Stratégies de gestion mémoire.

Travaux pratiques
Tests unitaires avancés. Optimisation des performances. Déploiement optimisé.

6 Gestion de la mémoire en C++

- Allocation dynamique (new, delete).
- Pointeurs, références, pointeurs intelligents (smart pointers).
- Fuites de mémoire, gestion des ressources (RAII).
- Bonnes pratiques de gestion mémoire.

Travaux pratiques
Manipulation de la mémoire. Atelier RAII et gestion des ressources. Optimisation mémoire.

7 Introduction à la STL (Standard Template Library)

- Présentation de la STL et de ses avantages.
- Conteneurs principaux : vector, list, map, set.
- Itérateurs et parcours de collections.
- Algorithmes standards (sort, find, etc.).

Travaux pratiques
Manipulation des conteneurs STL. Exercices sur les algorithmes STL. Optimisation et bonnes pratiques STL.

8 Patterns avancés et conception

- Patterns de conception classiques (Singleton, Factory, Observer, etc.).
- Utilisation des templates pour les patterns génériques.
- Bonnes pratiques de conception orientée objet.

Travaux pratiques
Implémentation de patterns. Patterns avancés et templates. Cas d'usage et revue de code.

9 Testing avancé et optimisation

- Tests unitaires avancés (mocks, tests paramétrés).
- Optimisation des performances (profiling, analyse de code).
- Stratégies de refactoring.

Travaux pratiques
Tests avancés. Optimisation et refactoring. Déploiement et synthèse.

10 Programmation générique et templates

- Templates de fonctions et de classes.
- Spécialisation et surcharge de templates.
- Templates variadiques et métaprogrammation de base.
- Bonnes pratiques et pièges à éviter.

Travaux pratiques
Création de templates. Métaprogrammation et templates avancés. Optimisation et bonnes pratiques.

11 Gestion des exceptions et robustesse

- Gestion des exceptions (try, catch, throw).
- Exceptions personnalisées.
- Bonnes pratiques de gestion d'erreurs.
- Impact sur la performance et la lisibilité.

Travaux pratiques
Manipulation des exceptions. Exercices sur la robustesse. Bonnes pratiques et revue de code.

12 Intégration de projets complexes

- Organisation d'un projet multifichiers.
- Utilisation de CMake ou d'autres outils de build.
- Gestion des dépendances et modularité.
- Documentation et tests automatisés.

Travaux pratiques
Structuration d'un projet. Intégration et gestion des dépendances. Cas d'usage et revue de projet.

13 Testing, CI/CD et synthèse

- Tests unitaires et d'intégration avancés.
- Introduction à l'intégration continue (CI/CD).
- Outils de build et de test automatisés.
- Synthèse des acquis de la journée.

Travaux pratiques
Mise en place de tests automatisés. CI/CD et automatisation. Synthèse et plan d'action.

14 Programmation avancée en C++

- Lambda expressions, auto, nullptr, move semantics.
- Boucles for-range, initialisation uniforme.
- Smart pointers avancés, gestion des ressources.
- Fonctions anonymes et closures.

Travaux pratiques
Exercices sur les nouveautés du langage. Ateliers sur la modernisation du code. Optimisation avancée.

15 Sécurité et robustesse en C++

- Sécurité mémoire (buffer overflow, use-after-free).
- Bonnes pratiques de validation des entrées.
- Gestion des accès concurrents (mutex, threads).
- Outils d'analyse de sécurité.

Travaux pratiques
Analyse de vulnérabilités. Exercices sur la concurrence. Bonnes pratiques et revue de code.

16 Performance et multithreading

- Introduction au multithreading en C++.
- Utilisation des threads, futures, promises.
- Synchronisation et gestion des ressources partagées.
- Outils de profiling et d'analyse de performance.

Travaux pratiques
Mise en œuvre du multithreading. Optimisation de la concurrence. Cas d'usage et revue de code.

17 Testing, monitoring et synthèse

- Tests de performance et de charge.
- Outils de monitoring (Valgrind, perf, etc.).
- Analyse des logs et détection d'anomalies.
- Synthèse des acquis de la journée.

Travaux pratiques
Tests de performance. Monitoring et analyse. Synthèse et plan d'action.

18 Projet de synthèse

- Analyse d'un cahier des charges.
- Conception orientée objet et modulaire.
- Développement d'une application C++ complète.
- Intégration des tests, optimisation et documentation.

Travaux pratiques
Réalisation du projet. Soutenance et retours.

19 Consolidation des bonnes pratiques

- Bonnes pratiques de développement C++.
- Gestion des erreurs et exceptions.
- Documentation technique et utilisateur.
- Planification de la maintenance et de l'évolution.

Travaux pratiques
Revue de code croisée. Atelier documentation et maintenance. Synthèse et bonnes pratiques.

20 Plan d'action personnel et clôture

- Définition d'objectifs personnels.
- Identification des ressources et outils pour progresser.
- Planification de la mise en pratique.
- Évaluation à chaud et feedback.

Travaux pratiques
Élaboration du plan d'action personnel. Évaluation et feedback. Clôture et perspectives.

21 UML, apprendre à modéliser avec les diagrammes - Contenu digital learning post-formation

- Notions fondamentales.
- Diagrammes structurels.
- Diagrammes comportementaux.

Activités digitales
Cette formation en ligne présente les fondamentaux de la conception orientée objet, les différents diagrammes UML, structurels et comportementaux, ainsi que leurs objectifs et leurs usages. À travers un exemple de conception fil rouge, l'application UML pour spécifier, visualiser et documenter efficacement un système informatique sera mise en pratique.

Dates et lieux

CLASSE À DISTANCE
2026 : 30 mars, 22 juin, 5 oct., 14 déc.

METZ
2026 : 22 juin, 14 déc.

PARIS LA DÉFENSE
2026 : 23 mars, 15 juin, 28 sep., 7 déc.

NANCY
2026 : 22 juin, 14 déc.