# C++, programmation objet

Référence : **DEC002** Durée : **5 jours** Certification : **Aucune**

## CONNAISSANCES PREALABLES

• Connaître les principes de la programmation orientée objet et disposer d'une expérience d'un langage de programmation.

## PROFIL DES STAGIAIRES

• Chefs de projets proches du développement. • Développeurs. • Ingénieurs.

## OBJECTIFS

• Appliquer les principes de la Conception Orientée Objet. • Maîtriser la syntaxe du langage C++. • Concevoir des applications C++ utilisant des classes. • Utiliser les outils de développement associés au langage C++. • Maîtriser les ajouts majeurs de la norme C++ 11 .

## CERTIFICATION PREPAREE

Aucune

## METHODES PEDAGOGIQUES

- Mise à disposition d'un poste de travail par stagiaire
- Remise d'une documentation pédagogique papier ou numérique pendant le stage
- La formation est constituée d'apports théoriques, d'exercices pratiques, de réflexions et de retours d'expérience
- Le suivi de cette formation donne lieu à la signature d'une feuille d'émargement

## FORMATEUR

Consultant-Formateur expert Développement C, C++

## METHODE D'EVALUATION DES ACQUIS

- Auto-évaluation des acquis par le stagiaire via un questionnaire
- Attestation de fin de stage adressée avec la facture

## CONTENU DU COURS

### Découverte du langage

- C++ depuis ses origines (langage C) jusqu'à nos jours (C++20)
- Syntaxe de base
- Fonctions et passage de paramètres
- Les tableaux et les enums
- Les types constants et les casts
- Copies, pointeurs et références
- Inférence de type avec le mot-clé auto
- Surcharge de fonctions et conversions implicites
- Créer des types utilisateur avec struct et union
- Définition d'alias avec typedef et using
- Gestion des erreurs avec les exceptions

### Structuration du code

- Les namespaces
- Déclaration et définition de symboles (One Definition Rule)
- Les fonctions et variables membres
- Les fonctions et variables statiques
- Principe de const correctness
- Les opérateurs et leur surcharge
- Les modules (C++20)

### Programmation orientée objet

- Classes, constructeurs et destructeurs
- Dérivation et héritage
- Visibilité et contrôle d'accès (public, protected, private)
- Mot-clé virtual et redéfinition de fonctions
- Polymorphisme et encapsulation
- Sémantique de valeur et sémantique d'entité
- Types abstraits et interfaces

- Principe de substitution de Liskov

## Gestion robuste des ressources logicielles

- Défis inhérents à la gestion des ressources
- Découverte du RAII
- Différence entre le tas et la pile
- Découverte de std::string et std::vector
- Portée, durée de vie et propriété (ownership)
- Découverte des pointeurs intelligents (smart pointers)
- Transfert de propriété (move semantic)

## Aperçu de la programmation générique et de la métaprogrammation

- Les différents types de polymorphisme
- Principes d'une fonction template
- Introduction aux classes template
- static_assert, constexpr et consteval (C++20)
- Découverte des templates variadic et du perfect forwarding
- Découverte de la métaprogrammation (SFINAE)
- Découverte des concepts (C++20)

## Design de code C++ moderne

- Introduction aux C++ Core Guidelines
- Notions de fonction pure et de testabilité du code
- Découverte de std::optional et std::variant
- Expressions lambda et std::function
- Type erasure avec std::string_view et std::span (C++20)
- Attributs standards : [[nodiscard]], [[deprecated]]
- La règle de zéro

## La bibliothèque standard

- Flux d'entrée/sortie (streams)
- Découverte de std::format() (C++20)
- Gestion du temps avec std::chrono
- Gestion de fichiers avec std::filesystem
- Principaux conteneurs de la STL
- Utilisation des algorithmes de la STL
- Programmation parallèle et concurrente (threads)
- Découverte des ranges (C++20)

## Construire un projet C++ avec CMake

- Utilité d'un build system tel que CMake
- Modes de compilation Debug et Release
- Découper son projet en plusieurs composants (bibliothèques)
- Intégrer des bibliothèques externes
- Ajouter et exécuter des tests unitaires
- Effectuer une compilation conditionnelle
- Les outils pour l'optimisation, le débogage, la vérification de code