C++, programmation avancée

3 j (21 heures) Ref : LANC++2

Public

Développeurs C++

Pré-requis

Avoir suivi la formation Langage C++

Moyens pédagogiques

Formation réalisée en présentiel ou à distance selon la formule retenue
Présentation des concepts, discussion technique, démonstrations, exercices simples et TP
Un poste informatique par stagiaire connecté à internet, à une imprimante en réseau et au réseau informatique
Les salles sont équipées d'un tableau interactif ou d'un vidéoprojecteur et d'un paperboard
Support de cours fourni à chaque stagiaire

Modalités de suivi et d'évaluation

Feuille de présence émargée par demi-journée par les stagiaires et le formateur
Exercices de mise en pratique ou quiz de connaissances tout au long de la formation permettant de mesurer la progression des stagiaires
Questionnaire d'évaluation de la satisfaction en fin de stage
Auto-évaluation des acquis de la formation par les stagiaires
Attestation de fin de formation

Boostez vos compétences en C++ avec cette formation avancée qui vous plongera au cœur des concepts modernes du langage.
Découvrez les dernières évolutions, optimisez vos performances avec le Move Semantics, exploitez la programmation concurrente et maîtrisez la métaprogrammation avancée.
Apprenez à concevoir un code robuste grâce à une gestion efficace des exceptions et des ressources, tout en intégrant les meilleures pratiques du Test Driven Development (TDD).
Grâce à une approche axée sur la pratique, vous manipulerez les templates avancés, les collections modernes, et les outils de la librairie standard pour développer des applications performantes et évolutives.

Objectifs

Maîtriser les mécanismes avancés de C++

Appréhender et mettre en oeuvre les techniques des nouvelles normes C++

Savoir pratiquer l'approche TDD (Test Driven Development) en C++

Découvrir et maîtriser la programmation parallèle et synchronisée

Programme détaillé

ÉVOLUTIONS DU LANGAGE

Concepts modernes du langage
Gestion de la mémoire et alignement
Inférence de types et déduction automatique
Nouveaux spécificateurs de classe et contrôle de l'héritage
Expressions constantes et évaluation à la compilation
Gestion avancée du temps et espace de noms chrono
Comparaisons avancées et opérateurs dédiés

MOVE SEMANTICS ET GESTION EFFICACE DES RESSOURCES

Différence entre copie et déplacement
std::move et références RValue
Constructeurs de déplacement et opérateurs d'affectation par déplacement
Interaction entre héritage et move semantics
Optimisation des structures de données avec swap et move
Mauvaises pratiques et pièges à éviter

GESTION AVANCEE DES RESSOURCES

Resource Acquisition Is Initialization (RAII)
Gestion de la mémoire avec unique_ptr, shared_ptr et weak_ptr
Stratégies avancées de gestion mémoire et custom deleter
Optimisation et allocation mémoire efficace

GESTION AVANCEE DES EXCEPTIONS

Hiérarchisation et conception des exceptions
Gestion avancée des erreurs et robustesse du code
Garanties de sécurité des exceptions
Usage avancé de noexcept et impacts sur l'optimisation

HERITAGE ET PROGRAMMATION ORIENTEE OBJET

Gestion avancée de l'héritage : privé, protégé, multiple
Héritage en diamant et héritage virtuel
Résolution des ambiguïtés et dominance
Interfaces et polymorphisme efficace

PROGRAMMATION FONCTIONNELLE EN C++

Utilisation avancée des lambdas et inline functions

Gestion des abonnements et callbacks

std::function, std::mem_fn et placeholders

Perfect forwarding et std::forward

decltype et inférence avancée des types

TEMPLATES, CONCEPTS ET METAPROGRAMMATION

Paramétrage avancé des templates

SFINAE (Substitution Failure Is Not An Error) et pattern matching

Variadic templates et passage d'arguments variable

Métaprogrammation et Curiously Recursive Template Pattern (CRTP)

Assertions statiques avec static_assert

Extensions avancées avec friend et typage générique

Contrats et concepts

UTILISATION AVANCEE DE LA LIBRAIRIE STANDARD

Collections modernes (std::array, unordered_map, unordered_set)

Structures avancées (tuple, listes simplement chaînées)

Itérateurs, adaptateurs d'itérateurs, Range

Algorithmes ensemblistes et traitements optimisés

MULTITHREADING ET PROGRAMMATION CONCURRENTE

Gestion avancée des threads et synchronisation

std::call_once, this_thread, async et futures

Mutexes, variables atomiques et gestion fine de la concurrence

Stratégies de synchronisation et meilleures pratiques

DEVOPS

TDD, approche et mise en place des tests unitaires

Gestion des dépendances et automatisation des tests

Frameworks et bonnes pratiques

Profiling et performances

Automatisation des Build

Développement Cross-Platform