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

Objectifs

Maîtriser les mécanismes avancés de C++
Appréhender et mettre en oeuvre les techniques de la nouvelle norme C++11/14
Savoir pratiquer l'approche TDD (Test Driven Development) en C++
Découvrir et maîtriser la programmation parallèle et synchronisée introduite par C++11

Programme détaillé

NOUVEAUTES DU LANGAGE C++11

nullptr_t et le littéral nullptr
Les nouveaux types, littéraux et séparateurs
Variables templates

- Initialisation uniforme
- Initialisation de tableaux et collections
- Parcours unifié des tableaux et conteneurs
- Listes d’initialisation avec initializer_listT>
- La boucle "range based" for
- Énumérations fortement typées
- Types normalisés et variantes (uint_8, uint64_t, ...)
- Contrôle de l’alignement mémoire
- Inférence de types et de signatures avec auto
- C++14 et déduction étendue
- Les nouveaux spécificateurs de classe (override, default, delete, final)
- Constructeur délégué
- Constructeur hérité
- Alias et using
- Expressions constantes avec constexpr
- Gestion du temps, l’espace de nom chrono

MOVE SEMANTICS

- Copie versus déplacement
- Value et RValue reference
- La fonction move
- Move constructor et move assignement operator
- Héritage et move constructor / op=
- STL C++11 et swap / move
- Signature reference qualifiers
- Mauvaises pratiques

GESTION DES RESSOURCES

- Resource Acquisition Is Initialization (RAII)
- L’opérateur -> avec ou sans généricité
- Propriété et transfert de responsabilité
- La classe unique_ptr
- Unique_ptr et tableaux dynamiques
- Comptage de références avec la classe shared_ptr
- Custom deleter
- Les fonctions make_unique (C++14) et make_shared
- La classe weak_ptr et le référencement circulaire

GESTION AVANCEE DES EXCEPTIONS

- Principes
- Dynamique
- Traiter une exception
- Concevoir et hiérarchiser les exceptions

Traitement par défaut

Les exceptions prédéfinies

Abraham’s Exception safety guarantees

La clause C++11 noexcept

HERITAGE AVANCE

Surcharge et ambiguïtés

Héritage public et redéfinition privée

Héritage privé et protégé

Héritage multiple

Héritage en diamant

Héritage virtuel et dominance

PROGRAMMATION FONCTIONNELLE AVEC C++ 11/14

Problématique de l’abonnement

Les classe function et mem_fn

Binding, placeholders

Adaptateurs de références

Les lambda-expression

UTILISATION AVANCEE

Typage multiple

Inférence des retours avec decltype

Paramétrage / spécialisation des méthodes

Perfect forwarding avec std:forward

Héritage / containment et généricité

Méta-programmation

L’idiome CRTP Curiously Recursive Template Pattern

Typologie C++ et classes de traits

Assertions statiques avec static_assert

Extended friend declaration

Les variadic templates, pattern matching et héritage multiple

SFINAE Substitution Failure Is Not An Error

NOUVEAUTES C++11 DE LA LIBRAIRIE STANDARD

Les nouveaux itérateurs cbegin

Les tableaux à taille fixe avec std ::array

Les nouvelles collections associatives : unordered_map/set

Singly-Linked Lists

Le conteneur tuple

Adaptateurs d’itérateurs, stream itérateurs

Les nouveaux algorithmes ensemblistes

MULTITHREADING

- Principes
- Démarrage et détachement d’un thread
- La classe std : call_once
- L’espace de noms this_thread
- Futures / promises et packaged_task
- Programmation asynchrone avec async
- Politiques de démarrage
- Comparaison thread versus future
- Partage de ressources et mécanismes de synchronisation
- Mutexes et données atomiques
- Unique-lock et lock_guard

APPROCHE TEST DRIVEN DESIGN (TDD) EN C++