



C++ Niveau 2

Mise à jour : Fév. 2024

Durée : 4 jours - 28 heures

OBJECTIFS PÉDAGOGIQUES

- Grace à cette formation vous pourrez
- Consolider vos connaissances dans le développement C++
- Disposer de connaissances pour rendre efficaces vos développements

PRÉREQUIS

- Il est nécessaire de
- Disposer de bonnes connaissances en développement C++.

PARTICIPANTS

- Concepteurs et développeurs d'applications en C++, chefs de projets, architectes logiciels.

MOYENS PÉDAGOGIQUES

- Réflexion de groupe et apports théoriques du formateur
- Travail d'échange avec les participants sous forme de réunion-discussion
- Utilisation de cas concrets issus de l'expérience professionnelle
- Validation des acquis par des questionnaires, des tests d'évaluation, des mises en situation et des jeux pédagogiques
- Remise d'un support de cours

MODALITÉS D'ÉVALUATION

- Feuille de présence signée en demi-journée
- Evaluation des acquis tout au long de la formation
- Questionnaire de satisfaction
- Attestation de stage à chaque apprenant
- Positionnement préalable oral ou écrit
- Evaluation formative tout au long de la formation
- Evaluation sommative faite par le formateur ou à l'aide des certifications disponibles

MOYENS TECHNIQUES EN PRÉSENTIEL

- Accueil des stagiaires dans une salle dédiée à la formation, équipée d'ordinateurs, d'un vidéo projecteur d'un tableau blanc et de paperboard

MOYENS TECHNIQUES DES CLASSES À DISTANCE

- A l'aide d'un logiciel comme Teams, Zoom etc... un micro et éventuellement une caméra pour l'apprenant,
- suivez une formation en temps réel et entièrement à distance. Lors de la classe en ligne, les apprenants interagissent et communiquent entre eux et avec le formateur
- Les formations en distanciel sont organisées en Inter-Entreprises comme en Intra-Entreprise
- L'accès à l'environnement d'apprentissage (support de cours, labs) ainsi qu'aux preuves de suivi et d'assiduité (émargement, évaluation) est assuré
- Les participants reçoivent une invitation avec un lien de connexion
- Pour toute question avant et pendant le parcours, une assistance technique et pédagogique est à disposition auprès de notre équipe par téléphone au 03 25 80 08 64 ou par mail à secretariat@feep-entreprises.fr

ORGANISATION

- Les cours ont lieu de 9h à 12h30 et de 13h30 à 17h

PROFIL FORMATEUR

- Nos formateurs sont des experts dans leurs domaines d'intervention
- Leur expérience de terrain et leurs qualités pédagogiques constituent un gage de qualité

ACCESSIBILITÉ

- Les personnes atteintes de handicap souhaitant suivre cette formation sont invitées à nous contacter directement, afin d'étudier ensemble les possibilités de suivre la formation. Notre organisme peut vous offrir des possibilités d'adaptation et/ou de compensations spécifiques si elles sont nécessaires à l'amélioration de vos apprentissages

Programme de formation

RAPPELS

- Classes d'allocation mémoire.
- Construction, initialisation, embarquement d'objets.
- Les fuites mémoire.
- Constance, le mot-clé mutable, Lazy Computation.
- Amitié (friendship) C++ et contrôle d'accès.
- Destruction virtuelle.
- Stratégie de gestion des exceptions.
- Les espaces de nommage (namespace).

LES NOUVEAUTÉS LANGAGE DE C++11

- nullptr et autres littéraux.
- Les directives =delete, =default.
- Délégation de constructeurs.
- Les énumérations "type safe".
- Le mot-clé auto et boucle sur un intervalle.
- Référence rvalue et impact sur la forme normale des classes C++.
- Les lambda expressions.

GESTION DES OPÉRATEURS

- Opérateurs binaires et unaires.
- L'opérateur d'indirection, cas d'usage.
- L'opérateur de référencement.
- Les opérateurs d'incrément/décément préfixés et post-fixés.
- Les autres opérateurs : comparaison, affectation...
- La surcharge de l'opérateur [], des opérateurs d'insertion (<<) et d'extraction (>>).
- Les foncteurs et la surcharge de l'opérateur (), avantage par rapport aux fonctions.

CONVERSION ET RTTI

- Opérateurs de conversion. Constructions implicites, le mot-clé explicit.
- Les opérateurs de casting const_cast, static_cast, reinterpret_cast.
- Conversion dynamique et Runtime Type Information.
- L'opérateur typeid, les exceptions liées.
- La classe typeid_info.
- Contrôle du "downcasting" à l'aide de l'opérateur dynamic_cast.

LA GÉNÉRICITÉ

- Introduction aux patrons de classe. Généricité et préprocesseur.
- Fonction générique. Classe générique. Composition générique. Généralisation générique.
- Spécialisation partielle et totale.
- Introduction à la méta-programmation.
- La généricité, principe fédérateur des bibliothèques STL et Boost.

LA STL (STANDARD TEMPLATE LIBRARY)

- Composants de la STL : types complémentaires, conteneurs, algorithmes, itérateurs, objets fonctions, les adaptateurs.
- Les chaînes de caractères STL, la classe template basic_string et ses spécialisations.

- Les conteneurs séquentiels et associatifs : définition, rôle et critères de choix.
- Les allocateurs et la gestion de la mémoire des conteneurs.
- Les méthodes d'insertion, de suppression, d'itération et d'accès aux principaux conteneurs : Vector, List, Set, Stack...
- Le concept d'itérateur. Parcours d'un conteneur.
- Les différents groupes d'algorithmes STL : non mutants, mutants, de tri et de fusion, numériques.
- Manipulation de conteneurs (manipulation, recherche de valeurs...).
- Paramétrer les algorithmes génériques par des objets "fonction".
- Les "adapteurs" et la modification du comportement d'un composant.
- La STL et les traitements sur les flux (fichiers, mémoire...).
- Principe du RAII : les pointeurs automatiques et la classe auto_ptr.
- Les exceptions standard de la STL.

LES NOUVEAUTÉS C++11 DE LA LIBRAIRIE STANDARD

- Evolution historique : Boost --> TR1 --> C++11.
- Les nouveaux conteneurs : array, forward_list, unordered_set, unordered_map.
- La classe tuple.
- Les pointeurs intelligents (smart pointer) : shared_ptr, weak_ptr, unique_ptr.
- Les nouveaux foncteurs et binders.
- Introduction à la gestion des threads.
- Les expressions régulières.

BOOST ET SES PRINCIPES

- La Pointer Container Library (destruction des données pointées d'un conteneur).
- Les structures de données boost::any et boost::variant.
- Programmation événementielle (connexions et signaux).
- Gestion des processus, mécanismes de communication interprocessus et mémoire partagée.

UTILISATION AVANCÉE DE L'HÉRITAGE

- Héritage versus embarquement. Héritage privé. Héritage protégé.
- Exportation de membres cachés avec la Clause Using.
- Héritage multiple et gestion des collisions de membres.
- Héritage en diamant. Héritage virtuel et dynamic_cast.
- Principes de conception : substitution de Liskov, principe d'ouverture/fermeture, inversion des dépendances.
- Règles d'implémentation des interfaces en C++.