Programmation C++, perfectionnement

Cours Pratique de 4 jours - 28h
Réf : POP - Prix 2024 : 2 390€ HT

En constante évolution, de C++98 à C++20, le langage C++ offre des mécanismes qui permettent une conception robuste et très riche. Les récentes normes C++ améliorent notablement la Standard Template Library (STL). Cette formation vous permettra d'approfondir la conception en C++ par l'apprentissage des dernières évolutions du langage et l'utilisation effective de la STL.

OBJECTIFS PÉDAGOGIQUES

À l'issue de la formation l'apprenant sera en mesure de :

Découvrir les nouveautés apportées par les versions

Maîtriser la gestion de la mémoire, des pointeurs et des références

Implémenter la généricité en C++

Découvrir la bibliothèque standard STL

Utiliser les apports de la norme C++11

TRAVAUX PRATIQUES

Le cours se déroulera sur des stations de travail sous Windows/Visual C++. De nombreux exercices permettront de mettre en oeuvre les thèmes abordés plus spécifiquement sous l'angle de la conception.

LE PROGRAMME

dernière mise à jour : 06/2021

1) 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).

2) 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.

Travaux pratiques : Réécriture d'un code C++ existant en C++11, comparaison des deux implémentations.

3) Gestion des opérateurs

- Opérateurs binaires et unaires.
- L'opérateur d'indirection, cas d'usage.
- L'opérateur de référencement.
- Les opérateurs d'incrémentation/décrémentation préfixés et post-fixés.

PARTICIPANTS

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

PRÉREQUIS

Bonnes connaissances en développement C++, ou connaissances équivalentes à celles apportées par le stage "Programmation Objet en C++" (réf. C++). Expérience requise.

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.

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 stages pratiques, études de cas ou présentation de cas réels pour les séminaires de formation.
• À l'issue de chaque stage 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 stagiaire 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

Vous avez un besoin spécifique d'accessibilité ? Contactez Mme FOSSE, référente handicap, à l'adresse suivante psh-accueil@orsys.fr pour étudier au mieux votre demande et sa faisabilité.

- 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.

Travaux pratiques : Création de foncteurs et de proxies (libération mémoire, comptage de références) avec les opérateurs étudiés.

4) 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 type_info.
- Contrôle du "downcasting" à l'aide de l'opérateur dynamic_cast.

Travaux pratiques : Mise en œuvre des idiomes "is-a" et "is-kind-of" avec dynamic_cast.

5) 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 librairies STL et Boost.

Travaux pratiques : Démarrage de l'étude de cas qui sera complétée avec la STL. Mise en œuvre de la composition et de la généralisation génériques. Création de plug-ins génériques.

6) 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.

Travaux pratiques : Implémentation des relations avec les collections de la STL. Utilisation d'algorithmes standard quelconques.

7) 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.

Travaux pratiques : Mise en œuvre de la robustesse avec les smart pointers. Utilisation d'expressions régulières.

8) 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.

Travaux pratiques : Amélioration de l'implémentation de l'étude de cas par l'utilisation la Pointer Container Library.

9) 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++.

Travaux pratiques : Combinaison de l'héritage multiple, privé et de l'exportation pour concevoir des classes robustes et hautement évolutives.

LES DATES

CLASSE À DISTANCE	PARIS
2024 : 25 juin, 01 oct., 26 nov.	2024 : 18 juin, 24 sept., 12 nov.