NIMBUS TRAINING
La formation NTIC • Tél : 02-40-85-19-32• cedric.tron@nimbus-training.fr

Modélisation UML et implémentation C++

Déscription :

La modélisation UML se base sur des pictogrammes qui permettent de modéliser tout au long du cycle de vie. A l'issue du stage, les participants seront capables de réaliser des spécifications UML appropriées aux applications industrielles et d'implémenter rigoureusement ces spécifications en langage C++ .

Objectifs

Être en mesure d'élaborer des diagrammes de classe et d'état transition rigoureux
- implémenter des classes et des association/agrégation/composition en langage C++
- implémenter des statecharts à l'aide d'un automate générique C++
- exploiter les bonnes pratiques C++ garantissant la production d'applications robustes et performantes

Publics

Développeurs en langage C

Durée

5 jours

Pré-requis

Cette formation s'adresse aux personnes possédant déjà une première expérience dans la mise en œuvre des concepts objet et le développement en langage C++.

Programme de cette formation

Le modèle objet
- Encapsulation, types abstraits
- Classes et instances
- Héritage de structure et d'interface
- Redéfinition et polymorphisme
- Généricité
- Illustration UML de tous les concepts (diagramme de classe et de séquence)

Les Classes et objets C++
- Définition d'attributs et de méthodes
- Propriétés publiques et privées
- Les différents constructeurs (défaut, paramétré, copie)
- Instanciation d'objets et portée (nommés, anonymes et dynamiques)
- Attributs et méthodes de classes (static)
- Les fonctions et classes amies (friend)

Les bonnes pratiques de l'encapsulation en C++

• Le destructeur et l'opérateur delete
• Surcharge de l'opérateur d'affection
• Surcharge des l'opérateurs de conversion sur les types prédéfinis
• Retour sur le constructeur par recopie
• Principe de la règles des trois
• Définition des namespaces

Les bonnes pratiques de l'héritage en C++

• Héritage public et héritage private
• Fonction virtuelle pure, classe abstraite et interface
• Héritage multiple et héritage virtuel
• Héritage versus composition
• Clonage d'objet
• Contrôle du typage à l'exécution
• Mécanismes RTTI

Transformations UML en langage C++

• Compléments sur les diagrammes de classes UML
• Mapping des propriétés d'une classe
• Mapping des relations (héritage et association)
• Collections permettant l'implantation des associations à cardinalité multiple
• Les diagrammes d'état transition UML
• Mapping des diagrammes d'état transition sous la forme d'une matrice
• Mapping des diagrammes d'état transition sous forme la forme d'un graphe d'héritage
• Exemple de génération de code C++ à partir d'AGL UML
• Reverse ingineering C++/UML

Les templates C++

• Objectifs des templates
• Template de classe
• Instanciation de templates
• Template de fonction
• Construction d'algorithmes génériques

LA STL (STANDARD TEMPLATE LIBRARY)

• Constituants de la STL
• La classe string
• Les conteneurs : vector, deque, list, map et itérateurs
• Allocateurs de conteneurs
• Les algorithmes et adaptateurs

Les bonnes pratiques liees à la robustesse et a la performance

• Les exceptions C++
• Les pointeurs « intelligents » : auto_ptr et smart pointer
• Surcharge des opérateurs new, delete
• Recyclage des objets et mise en pool
• Automate générique
• Interfaçage avec le langage C