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

Langage C++ perfectionnement

Déscription :

Une modélisation efficace nécéssite un langage en conséquence. Le langage C++ prouve régulièrement sa robutesse et la stabilité au niveau des applications. Cette formation permet de se perfectionner en C++

Objectifs

• développer des classes C++ fiables et performantes,
• élaborer des solutions évolutives en utilisant l'héritage, le polymorphisme et la généricité,
• exploiter ses propres pools de mémoire via la surcharge des opérateurs standards,
• exploiter de manière efficace les composants de la STL,
• organiser la robustesse autour des exceptions,
• savoir se prémunir des fuites mémoires en exploitant les pointeurs intelligents de TR1,
• connaître les fonctionnalités proposées par la librairie boost.

Publics

Personnes amenées à développer des applications en langage C++

Durée

4 jours

Pré-requis

Personnes ayant une première expérience de développement en langage C++

Programme de cette formation

Implémentation rigoureuse des classes C++

• Rappel sur les différents types de constructeurs
• L'intérêt des listes d'initialisation
• Forme canonique de Coplien
• Les propriétés statiques, utilisation des « friends »
• Les attributs « const » et « inline »
• Organisation des sources avec les espaces de nommage (namespace)

Bonnes pratiques de l'Héritage en C++

• Héritage public versus héritage private
• Héritage versus composition
• Constructeur, destructeur et virtualité
• Héritage multiple et héritage virtuelle

• Règles d'implantation des interfaces en C++
• Ambiguïté potentielle et résolution par la clause using

Conversion et RTTI

• Opérateurs de conversion : constructions implicites et mot-clé explicit
• Les nouveaux opérateurs de casting : const_cast, static_cast, reinterpret_cast
• Conversion dynamique et RTTI
• La structure type_info et l 'opérateur dynamic_cast

surcharge des opérateurs, aspects avancés

• Introduction aux pointeurs intelligents, surcharge des opérateurs ++, --, * et ->
• Implantation d'un itérateur, classe interne
• Insertion/extraction d'objets sur des flux, surcharge d'opérateurs <<, >>
• Les foncteurs et la surcharge de l'opérateur ()

GESTION AVANCEE DE LA MEMOIRE EN C++

• Le comportement standard des opérateurs new et delete, new[] et delete[]
• Limites de l'allocation dynamique standard
• Pool d'allocation mémoire spécifique
• Surcharge des opérateurs new et delete

Les templates C++

• Objectifs des templates
• Template de classe et de fonction
• Instanciation de templates, spécialisation partielle ou complète
• Introduction à la méta programmation en C++

La STL (Standard Template Library)

• Organisation de la STL
• Classes utilitaires : string, complex, pair, auto_ptr, …
• Les conteneurs : vector, deque, list, map et itérateurs
• Modification de l'allocateur de conteneur
• Les algorithmes et les adaptateurs

Bonnes pratiques des exceptions C++

• Stratégies de traitements d'erreurs
• Levée et capture d'exceptions : opérateur throw et les blocs try/catch
• Exceptions et constructeurs
• Exceptions standards STL
• Fonctions terminate et unexpected

Introduction à TR1 et à la librairie BOOST

• TR1, boost et la nouvelle norme C++ (C++11)
• Le pointeurs « intelligents » de TR1 : shared_ptr, weak_ptr et unique_ptr
• Les nouvelles collections de TR1 : unordered_map et unordered_set
• Les structures de données de boost : tuple, any et variant
• Tour d'horizon des autres fonctionnalités de boost