Langage C++

Programmation avancée

Objectifs : Maîtriser la programmation avancée en C++, pouvoir réaliser des applications performantes et bien structurées, découvrir les principales fonctionnalités offertes par la bibliothèque standard STL. **Prérequis** : Avoir suivi la formation « Langage C++ - Programmation » ou avoir les connaissances équivalentes. **Public** : Développeurs C++.	
Niveau : Perfectionnement	**Durée standard préconisée** : 30h
Pédagogie : alternance d'apports théoriques et nombreux exercices de mise en pratique	
Moyens pédagogiques : un ordinateur multimédia par apprenant, ordinateur et vidéoprojecteur pour l'animateur	**Évaluation des acquis** : Mise en pratique à l'aide d'exercices en autonomie puis corrigés individuellement et collectivement
Suivi après formation : • Certificat de stage • Un ouvrage de référence (remis en formation)	

Détail des objectifs

Maîtriser et approfondir les éléments clés
- Maîtriser la gestion de la mémoire
- Maîtriser le mécanisme de destruction virtuelle
- Maîtriser la construction de tableau d'objets, l'opérateur delete
- Maîtriser la gestion de surcharge des opérateurs new, delete et delete
- Maîtriser la gestion des foncteurs (pour surcharge des opérateurs)
- Maîtriser la gestion de surcharge des opérateurs de conversion operator char…
- Maîtriser les pointeurs sur les fonctions
- Maîtriser les gestions des exceptions et les hiérarchiser

Utiliser les espaces de nom
- Définir un espace de nom (namespace)
- Utiliser la clause using

Appréhender RTTI (Run-Time Type Information)
- Comprendre typeid et type_info
- Gérer les opérateurs de « cast » const_cast, static_cast, reinterpret_cast
- Gérer le downcasting avec l'opérateur dynamic_cast

Maîtriser la notion de généricité
- Construire une fonction et une classe générique
- Utiliser un foncteur et pointeur sur fonction en paramètre d'une classe générique
- Utiliser l'héritage pour spécialiser une classe générique

Intégrer la bibliothèque STL (Standard Template Library)
- Comprendre l'architecture de la STL : conteneurs, itérateurs, allocateurs et algorithmes
- Utiliser un conteneur élémentaire, la classe vector
- Utiliser les autres conteneurs de base, les listes, les DQ
- Utiliser les conteneurs associatifs
- Utiliser les itérateurs, les itérateurs de flux
- Utiliser les adaptateurs pour modifier le comportement d'un composant
- Utiliser les algorithmes
- Gérer des exceptions de la STL
- Utiliser un pointeur intelligent (« Smart pointer ») avec la classe auto_ptr

Intégrer BOOST
- Définir ce qu'est « Boost », et sa normalisation avec tr1
- Utiliser les principales bibliothèques : les threads, les expressions régulières
- Utiliser la méta-programmation
- Gérer la mémoire avec des pointeurs intelligents (Smarts pointers)…
- Programmer des threads
- Utiliser les « Smart pointers » de boost, shared_ptr, weak_ptr, unique_ptr