

# Formation Maîtriser le langage C++



La mise en pratique des concepts de modélisation Objet nécessite des langages de programmation adaptés. Le langage C++, digne successeur du langage C, fût l'un des premiers acteurs à voir le jour sur le marché du développement. Depuis une vingtaine d'années, ce langage a prouvé à de très nombreuses reprises son fort potentiel au niveau de la performance et de la stabilité des applications

## Objectifs

- Acquérir les principes de base de la programmation Objet (polymorphisme, héritage, encapsulation)
- Maîtriser le langage C++
- Apprendre à réaliser des applications réutilisables
- Connaître les meilleures pratiques de tests de qualité en C++

## Public concerné

- Développeurs
- Concepteurs

## Pré requis

- Connaissance du langage C ou expérience d'un langage de programmation structurée

## Une formation de 5 jours

Caractéristiques	Paris
Tarif : 2225 € HT par personne	04/04/2011
Numéro de formateur : 11753687675	20/06/2011
Nombre d'heures : 35	26/09/2011
Référence : LA200	21/11/2011
Contact : Patrick LE GOFF	
Telephone : 01.76.60.66.10	
Email : <a href="mailto:contact@kaptive.com">contact@kaptive.com</a>	

## Description des modules

num	Module
<b>1</b>	<b>Du procédural à l'objet</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Bref historique des langages</li> <li>- Critères de qualité dans un développement logiciel</li> <li>- Aperçu général des langages orientés Objet</li> </ul>
<b>2</b>	<b>Les concepts objet</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Classes, objets (constructeurs et paramètres par défaut)</li> <li>- Encapsulation (visibilité public-private-protected, namespaces)</li> <li>- Membres et méthodes de classes (static)</li> <li>- Héritage simple</li> <li>- Héritage multiple (classe de base virtuelle)</li> <li>- Polymorphisme (virtuel), Classes abstraites (virtuel pures)</li> <li>- Interfaces</li> </ul>
<b>3</b>	<b>Syntaxe C++</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Fonctions "Friends"</li> <li>- Fonctions "Inline"</li> <li>- Paramètre caché this</li> <li>- Point d'entrée main avec arguments</li> <li>- Type références</li> <li>- Classes imbriquées (Inner class)</li> <li>- Inclusion des headers et références multiples</li> <li>- Utilisation du qualificateur const</li> <li>- Bibliothèque standard d'E/S</li> <li>- Gestion dynamique de la mémoire</li> <li>- Destructeurs virtuels</li> <li>- Typage dynamique avec RTTI</li> </ul>
<b>4</b>	<b>Traitement des exceptions</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Traitement des erreurs dans les programmes</li> <li>- Traitement des erreurs en C++</li> <li>- Traitement des exceptions imbriquées</li> <li>- Classes d'exception applicatives</li> </ul>
<b>5</b>	<b>Surcharge</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Surcharge des fonctions</li> <li>- Surcharge d'opérateurs</li> <li>- Surcharge de l'opérateur new (avec nothrow) et set-new_handler</li> </ul>
<b>6</b>	<b>Patrons - "Templates"</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Définitions de patrons, syntaxe et instanciation</li> <li>- Patrons de fonctions</li> <li>- Template de classes (exemple avec la classe smart pointer)</li> </ul>
<b>7</b>	<b>Points clé de la STL</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Présentation des principaux conteneurs (vector, list, set, map, deque)</li> <li>- Critères de choix pour un conteneur STL</li> <li>- Les itérateurs</li> <li>- Les algorithmes génériques</li> </ul>
<b>8</b>	<b>Introduction aux Design Pattern</b>
<b>Détails</b>	<ul style="list-style-type: none"> <li>- Pattern singleton</li> <li>- Modèle observateur MVC avec exemple d'implémentation en C++</li> </ul>

## 9 **Qualité logicielle**

- Détails**
- Les best practices en C++
  - Règles de conception et astuces de codage
  - Conclusion avec comparatif Java / C++