

Stage pratique de 4 jour(s)  
Réf : POP

### Pré-requis

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

Prix 2012 : 2150€ HT

### Dates des sessions

#### Paris

27 mar. 2012, 26 juin 2012  
18 sep. 2012, 13 nov. 2012

# Programmation C++, perfectionnement

## OBJECTIFS

*En constante évolution, le langage C++ offre des mécanismes tels que la généricité ou la méta-programmation permettent une conception robuste et très riche. Les récentes normes Technical Report TR1 et TR2 déjà intégrées dans C++ reprennent l'essentiel des bibliothèques du projet BOOST qui améliorent notablement la Standard Template Library (STL). Cette formation vous permettra d'approfondir la conception en C++ par l'apprentissage de très nombreux mécanismes et l'utilisation effective de la STL.*

### 1) Rappels

### 2) Gestion des opérateurs

### 3) Conversion et RTTI

### 4) La généricité

### 5) La STL (Standard Template Library)

### 6) BOOST

### 7) Utilisation avancée de l'héritage

## 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.*

## 1) Rappels

- Classes d'allocation mémoire. Construction, initialisation, embarquement d'objets. Constance.
- Amitié (friendship) C++ et contrôle d'accès. Destruction virtuelle. Gestion des exceptions.

## 2) Gestion des opérateurs

- Opérateurs binaires et unaires.
- Les foncteurs. L'opérateur d'indirection.
- Opérateurs d'incrément/décrément et d'autres opérateurs.

### Travaux pratiques

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

## 3) 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 RTTI (Runtime Type Information).
- La structure `type_info`. L'opérateur `dynamic_cast`.

### Travaux pratiques

*Mise en oeuvre des idiomes " is-a " et " is-kind-of " avec `dynamic_cast`.*

## 4) La généricité

- Introduction aux patrons de classe. Généricité et préprocesseur.
- Fonction générique. Classe générique. Composition générique. Généralisation générique.
- 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 et BOOST. Mise en oeuvre de la composition et de la généralisation génériques. Création de plug-ins génériques.*

## 5) La STL (Standard Template Library)

- Introduction. Les conteneurs séquentiels et associatifs : Définition, rôle et critères de choix.
- Le concept d'itérateur. Parcours d'un conteneur. La classe `auto_ptr`.
- Algorithmes STL et manipulation de conteneurs (manipulation, recherche de valeurs...).
- La STL et les traitements sur les flux (fichiers, mémoire, ...).

### Travaux pratiques

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

## 6) BOOST

- Présentation de Boost. Les nouveaux conteneurs.
- Les pointeurs intelligents (smart pointers). La Pointer Container Library (destruction des données pointées d'un conteneur).
- Foncteurs, binders et lambda-expressions.
- Programmation événementielle (connexions et signaux). Introduction à la gestion des threads.
- Gestion des processus, mécanismes de communication interprocessus et mémoire partagée.
- Les structures de données (Tuple, Any, Variant).

### Travaux pratiques

*Mise en oeuvre de la robustesse avec les smart pointers de BOOST.*

## 7) 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.

### **Travaux pratiques**

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