

Formation C++, Apprendre la programmation orientée objet

Présentation

En cinq jours (35 h), ce module « C++ : programmation Objet » conduit les développeurs à pratiquer un C++ moderne et robuste : de la syntaxe de base aux fonctionnalités post-C++20, en passant par l'encapsulation, l'héritage, les modèles génériques et la gestion sûre de la mémoire. Chaque journée se termine par un mini-atelier adaptable afin d'ancrer immédiatement les notions vues.

Durée : 35,00 heures (5 jours)

Tarif INTRA : [Nous consulter](#)

Objectifs de la formation

A l'issue de la formation, le stagiaire sera capable de mettre en œuvre les principes fondamentaux de la conception orientée objet et de concevoir des applications en C++

- Comprendre la syntaxe et les concepts fondamentaux du C++
- Maîtriser les ajouts majeurs des normes C++
- Appliquer les principes de la Conception Orientée Objet
- Écrire des programmes simples en appliquant les bonnes pratiques de développement
- Utiliser les structures de contrôle et les types de données en C++
- Manipuler les fichiers et la mémoire de manière basique

Prérequis

Connaître les principes de la programmation orientée objet et disposer d'une expérience d'un langage de programmation...

Public

Développeurs, ingénieurs, chefs de projets proches du développement

Programme de la formation



Jour 1 – Fondamentaux du langage

- Comprendre le cycle écrire → compiler → exécuter
- Manipuler types, portées et contrôles de flux en C++ moderne

Syntaxe de base

- Variables, constantes
- Portée des noms
- Déduction de type automatique
- Conditions (if, switch sûr) et boucles classiques

Modèle de compilation C/C++

- Rôle des fichiers en-tête / implémentation
- Impact sur le temps de build

Bonnes pratiques initiales

- Nommage clair
- Commentaires Doxygen
- Compilation avec options avertissement strict

Atelier pratique

- Créer un petit programme console (par ex. change d'unités, devises, longueurs...)
- Compiler, exécuter et identifier les fichiers générés par le compilateur

Jour 2 – Classes, encapsulation & RAII

- Concevoir une classe sûre et lisible
- Gérer automatiquement la durée de vie des ressources

Conception de classe

- Attributs privés

- Fonctions d'accès
- Constantes d'instance

Cycle de vie RAI

- Constructeur pour acquérir, destructeur pour libérer
- Copier ou déplacer : choisir selon le besoin
- Std::move

Espaces de noms & modules

- Accélérer la compilation
- Éviter les collisions

Atelier pratique

- Transformer un code « ouverture/fermeture » manuel (fichier, socket, mutex, etc.) en une classe qui gère automatiquement la ressource

Jour 3 – Héritage, polymorphisme & STL de base

- Appliquer héritage public et polymorphisme d'exécution
- Utiliser les conteneurs et algorithmes STL essentiels

Héritage & interfaces

- Méthodes virtuelles
- Classes abstraites
- Mots-clés override/final

Polymorphisme d'exécution

- Activer la bonne version d'une méthode
- Sécuriser la hiérarchie avec les mots-clés

Gabarits (fonctions génériques)

- Écrire une seule fonction utilisable avec plusieurs types compatibles

Design patterns simples

- Factory, Strategy...

Collections standard (aperçu)

- Tableau extensible, liste chaînée
- Dictionnaire clé-valeur : finalités et complexité
- Parcourir les collections

Atelier pratique

- Définir une classe de base (par ex. Employé, Forme, Capteur) puis deux dérivés, stockés dans une collection standard et parcourus pour afficher une information clé

Jour 4 – C++ moderne & gestion mémoire avancée

- Gérer la mémoire sans fuite
- Écrire un code concis avec fonctions anonymes et algorithmes standard
- Traiter les erreurs d'exécution de façon centralisée

Pointeurs intelligents

- Propriété exclusive, partagée
- Quand choisir l'un ou l'autre ?
- Utiliser `std::optional`, `std::variant`, `std::any`

Fonctions anonymes (lambdas)

- Créer un petit calcul "à la volée"
- Capturer des variables par copie ou référence

Algorithmes standards

- Trier, transformer
- Additionner des éléments sans boucle manuelle
- Gammes de vues en C++20

Calcul à la compilation

- constexpr / constexpr / constexpr

Gestion propre des erreurs

- Lever / attraper des exceptions

Atelier pratique

- Remplacer des pointeurs « nus » par des pointeurs intelligents dans un petit tableau d'objets, puis trier ce tableau selon plusieurs critères grâce à une fonction anonyme

Jour 5 – Fichiers, optimisation & mise en production

- Lire et écrire des données texte ou binaires sans risque de corruption
- Améliorer les performances grâce au profilage et aux options d'optimisation
- Préparer un projet réutilisable, testé et versionné pour l'équipe

Entrées / sorties fichiers

- Ouvrir un fichier en lecture ou écriture, choisir texte vs binaire
- Sauvegarder / charger un objet dans un format clair ou compact

Profilage & optimisation

- Mesurer où le programme passe le plus de temps CPU ou mémoire
- Activer les optimisations du compilateur
- Vérifier le gain obtenu

Construction et tests

- Créer un script de compilation multi-plateforme
- Écrire un test automatique
- Catch2 ou GoogleTest

Packaging & versions

- Produire une bibliothèque statique ou dynamique
- Produire les fichiers d'en-tête publics
- Appliquer une numérotation claire pour les livraisons

Atelier pratique

- Sur la base d'un module existant, l'intégrer dans un projet de construction (outil standard)
- Ajouter un test automatique minimal
- Produire un paquet prêt à être partagé en interne

Organisation

Formateur

Les formateurs de Docaposte Institute sont des experts de leur domaine, disposant d'une expérience terrain qu'ils enrichissent continuellement. Leurs connaissances techniques et pédagogiques sont rigoureusement validées en amont par nos référents internes. Riches de leur expérience sur le sujet, ils sauront accompagner vos collaborateurs dans leur montée en compétences.

Moyens pédagogiques et techniques

- Alternance d'apports théoriques et de mises en pratique contextualisées.
- Progression pédagogique structurée, de la compréhension à la mise en œuvre opérationnelle.
- Pédagogie immersive favorisant une mise en application immédiate des compétences.
- Formation interactive et participative s'appuyant sur des ateliers pratiques, études de cas et échanges collectifs (+ 60 % du temps de formation.)

- Séquences pédagogiques rythmées et évaluations formatives régulières pour renforcer l'ancrage des apprentissages.
- Supports de cours, ressources documentaires et références mis à disposition par le formateur

Dispositif de suivi de l'exécution et de l'évaluation des résultats de la formation

En amont de la formation (*évaluation en ligne dans l'extranet apprenant*)

- Recueil des besoins et des attentes des participants.
- Auto-positionnement permettant d'évaluer le niveau initial des apprenants.

Tout au long de la formation

- Évaluations continues réalisées par le formateur travers des questions orales, exercices, QCM, études de cas et mises en situation pratiques.

À l'issue de la formation (*évaluations en ligne dans l'extranet apprenant*)

- Auto-positionnement afin de mesurer l'acquisition des compétences visées.
- Validation de l'acquisition des compétences par le formateur
- Questionnaire de satisfaction « à chaud » à l'issue de la formation
- Questionnaire de satisfaction "à froid", 45 jours après la formation afin d'évaluer l'ancrage des acquis et leur mise en application en situation professionnelle.

Accessibilité

Nos formations peuvent être adaptées à certaines conditions de handicap. Nous contacter pour toute information et demande spécifique.