C++ Certified Programmer – CPP

Objectifs

- Concevoir l'architecture d'une application C++ en utilisant ses propres classes ou des classes existantes
- Savoir programmer objet en C++ (classe, méthode, propriétés, constructeur, instance, etc.)
- Maîtriser la syntaxe du C++
- Maîtriser son environnement de développements (outils, compilation, build, etc.)
- Connaître la norme C++ 11 et les nouveautés introduites par cette version et les suivantes

Prérequis

- Cette formation C++ demande de savoir développer dans au moins un langage de programmation (C, PHP, Java, C#, Python, etc.) car la formation ne prévoit pas de revenir sur les bases (variables, test, boucle, appel fonction, etc.).
- Concernant la dimension objet il est conseillé de disposer d'une culture objet sur les fondamentaux (classe, propriétés, méthodes, instance, héritage) et de ne pas totalement les découvrir même si la formation revient dessus la première demi-journée dans le cadre du C++.

Programme

J1

Les concepts de la Programmation objet essentiels pour bien commencer en C++

Les paradigmes de la P.O.O.
La classification
Contrôler l'accès aux données (encapsulation)
Introduction aux diagrammes de classes UML2
Les associations entre classes : l'association directe, l'agrégation, la composition
L'héritage, la dérivation
Fournir les informations essentielles (abstraction)Travaux pratiques (à titre indicatif)***Objectifs*** *: Valider que les concepts objets fondamentaux sont maîtrisés par tout le monde*
Présenter le modèle objet que l'on va utiliser en fil rouge dans la formation (il ne sera pas unique afin de bien comprendre les manipulations communes à toutes les classes et ce qui est spécifique à chacune)
Description *: Quizz qui valide dans un premier temps que tout le monde dispose du vocabulaire objet, que ce vocabulaire est commun à tous participants et au formateur, mais aussi que tout le monde a compris en profondeur chaque concept.*

C++, le langage

Les types de données, les opérateurs
Les pointeurs bruts
Le type référence lvalue
Les structures de contrôles
Les fonctions et la surcharge
L'opérateur de résolution de portée
Liaison C - C++
Les paramètres par défaut des fonctions
Les fonctions " inline "
Les espaces de noms
Les énumérationsTravaux pratiques (à titre indicatif)**Objectifs** : Valider que tout le monde est à l'aise avec son environnement de développement et prend en main la syntaxe de base du C++ (opérateurs, types, tests, etc.).
Description : Réalisation du programme types.cpp, qui définit et initialise une variable de chacun des types primitifs disponibles (bool, char, int, etc…). Chacune des variables est ensuite affichée en utilisant l'objet cout
Saisie d'un nombre décimal, calcul, affichage du résultat
Réalisation du programme celcius.cpp qui demande la saisie au clavier d'une température en degrés Fahrenheit, puis calcule et affiche la température correspondante en degrés Celsius en interdisant les valeurs hors plage.J2

Les classes en C++

Définition d'une classe
Les qualificateurs " public " et " private "
Instanciation d'une classe
Les constructeurs par défaut
Les constructeurs surchargés
Le constructeur de copie
Surcharge de l'opérateur d'affectation
Le destructeur
Le mot-clé this
Les méthodes const
Les membres static
Fonctions amies
Surcharges d'opérateursTravaux Pratiques (à titre indicatif)**Objectifs** : Maîtriser la création de classe, l'instanciation, les constructeurs et la surcharge des méthodes
Description : Création d'un classe « counter » permettant de créer un objet qui génère une séquence de nombres (incrémentation, valeur actuelle, affichage, etc.)
Surcharges du constructeur en y passant un nombre ou un objet (pour s'initialiser sur cet objet)J3

Relations entre classe (association, composition, héritage)

Traduction en C++ d'une association, d'une composition
Importance de la liste d'initialisation dans les constructeurs
L'héritage public simple
Ordre d'exécution des constructeurs et destructeurs

Les règles de conversion
Le qualificateur d'accès protected
Les héritages privé et protégé
L'héritage multiple répété et ses difficultés de mise en oeuvreTravaux Pratiques (à titre indicatif)**Objectifs** : Comprendre comment mettre en œuvre une association directe, puis une composition entre classes. Savoir spécialiser une classe existante par héritage pour ne pas tout réécrire en maîtrisant l'impact hiérarchique dans la syntaxe (classe mère et descendance.)
Description : Mise en œuvre d'une relation d'association puis de composition entre deux classes. Construction d'une hiérarchie de classes à trois niveaux. Manipulation d'objets de ces classes. Mise en œuvre d'un héritage multiple répété

Le polymorphisme

Les méthodes virtuelles
Intérêt du polymorphisme
Mise en œuvre du polymorphisme
Les classes abstraites et les méthodes virtuelles pures
Les destructeurs virtuelsTravaux Pratiques (à titre indicatif)**Objectifs** : Savoir mettre en œuvre le polymorphisme,
Description : Mise en œuvre du polymorphisme dans le Design Pattern Factory Method : obtenir un objet d'une sous-classe dans une hiérarchie de classes avec héritage multiple comportant une classe abstraite pureJ4

La gestion des exceptions

Principe du traitement des anomalies
Utilisation des mot-clés try, catch et throw
Traiter les exceptions sur place
Propager les exceptions
Lever une exception
Créer ses propres classes d'exceptionTravaux Pratiques (à titre indicatif)**Objectifs** : comprendre comment gérer les exceptions dans une application effectuant des entrées-sorties
Description : Calcul de la somme des soldes mensuels d'un compte bancaire par lecture d'un fichier CSV avec mise en œuvre d'un traitement sur place d'exceptions puis en propageant les exceptions à la fonction appelante

Les templates en C++

Principe des templates
Avantages/inconvénients
Paramètres template
Syntaxe des templates de fonctions
Syntaxe des templates de classes
Syntaxe des templates de méthodes
Instanciation des templates de fonctions
Instanciation des templates de classes
Instanciation implicite vs instanciation expliciteTravaux Pratiques (à titre indicatif)**Objectifs** : Comprendre et mettre en œuvre la généricité des types dans une fonction grâce aux templates
Description : Création d'un template de fonction (fonction rendant le « plus petit » de 2 objets

passés en argument indépendamment de leur type). Création d'un template de classe (tableau redimensionnable automatiquement)J5

Les pointeurs intelligents (smart pointers C++11)

Principe de la gestion RAII des ressources
Danger des pointeurs bruts
Intérêt des smart pointers
Raisons de l'obsolescence de auto_ptr
Mise en œuvre de unique_ptr, shared_ptr, weak_ptr
Utilisation de make_unique et make_sharedTravaux Pratiques (à titre indicatif)**Objectifs** : Comprendre la mise en œuvre des smart pointers pour la gestion des ressources
Description : Mise en oeuvre, dans une petite application, de la technique RAII par remplacement des pointeurs bruts par des pointeurs intelligents

Outils de développement

Présentation des IDE courants
Options de compilation
Cross-compilateurs
Génération d'un exécutable, fichiers objets
Utilisation des fonctionnalités de debug
Commande make sous Linux, makefile
Utilisation de gitTravaux Pratiques (à titre indicatif)**Objectifs** : Savoir debugguer une application et modifier les options de compilation. Savoir packager une application C++.
Description : Recherche de bugs dans une application et correction. Mise en œuvre d'une chaîne de build automatisée.

Aperçu de la librairie STL

Présentation, documentation
Les conteneurs
Les itérateurs
Les algorithmes
Les entrées-sorties
L'espace de noms chronoDémonstration (à titre indicatif)**Objectifs** : comprendre l'intérêt d'un conteneur pour y stocker des objets et le rôle d'un itérateur. Savoir lire-écrire dans un fichier texte
Description : lecture d'un fichier texte ligne par ligne, création des objets représentés par les chaînes de caractères, mise en œuvre d'un vector pour y stocker les objets et parcours à l'aide d'un itérateur pour supprimer certains objets, tri du vector à l'aide d'un algorithme de tri et d'un foncteur

Présentation des principales nouveautés apportées par C++11/14/17/20

Mot-clé auto
Boucle for-each
Le mot-clé noexcept
Références rvalue et applications
Expressions lambdas et applications

Initialisation uniforme
Méthodes par défaut et supprimées
Concepts et contraintes
ModulesDémonstration (à titre indicatif)**Objectifs** : comprendre les simplifications apportées par le mot-clé auto, la boucle for-each et les expressions lambdas
Description : ré-écriture de l'exemple précédent en utilisant une boucle for-each et auto pour afficher le contenu du vector, utilisation d'une expression lambda dans l'algorithme de tri