Présentation de la certification LabVIEW

Le programme de certification LabVIEW de National Instruments est composé des trois niveaux de certification suivants :

- Examen CLAD (Certified LabVIEW Associate Developer)
- Examen CLD (Certified LabVIEW Developer)
- Examen CLA (Certified LabVIEW Architect)

Chaque niveau de certification est prérequis pour passer au niveau suivant.

L'obtention du CLAD démontre que vous avez une compréhension étendue et complète des fonctionnalités principales disponibles dans le système de développement complet de LabVIEW et que vous êtes capable d'appliquer vos connaissances pour développer, mettre au point et maintenir des petits modules LabVIEW. Le niveau d'expérience typique pour le CLAD est d'environ 6 à 9 mois d'utilisation du système de développement complet de LabVIEW.

L'obtention du CLD démontre de l'expérience dans le développement, la mise au point, le déploiement et la maintenance d'applications LabVIEW de moyenne ou grande envergure. Le niveau typique pour un professionnel qui passe le CLD est une expérience cumulée de 12 à 18 mois dans le développement d'applications LabVIEW de moyenne ou grande envergure.

L'obtention du CLA démontre que vous possédez l'expertise technique et l'expertise de développement de logiciel nécessaires pour décomposer une spécification de projet en composants LabVIEW gérables, et que vous êtes capable de mener à bien un projet en utilisant de manière efficace les outils de gestion de configuration et de projet. Le niveau typique pour un professionnel qui passe le CLA est une expérience cumulée d'environ 24 mois dans le développement d'applications LabVIEW de moyenne ou grande envergure.

Remarque La certification CLAD est prérequise pour passer le CLD. La certification CLD est prérequise pour passer le CLA. Il n'y a pas d'exception à la règle pour chaque examen.

Présentation de l'examen CLA

L'obtention du CLA démontre la maîtrise de l'analyse et de l'interprétation de spécifications de clients pour le développement, dans LabVIEW, d'architectures d'applications extensibles, organisées en hiérarchie de projet modulaire, en vue d'un développement complet ultérieur par une équipe de développeurs. L'architecture remplit les spécifications avec une simulation logicielle qui utilise des éléments servant d'interfaces avec des modules matériels abstraits. Pour assurer le succès de l'intégration, l'Architecte permet à une équipe de développeurs de terminer l'application en lui fournissant des modules conçus avec des interfaces, des structures de données, des modèles de conception de modules et une communication basée sur des messages, le tout cohérent et bien défini, ainsi que des instructions documentées avec des spécifications de conception spécifiques. Le niveau typique pour un professionnel qui passe le CLA est une expérience cumulée d'environ 24 mois dans le développement d'applications LabVIEW de moyenne ou grande envergure.

Produit : L'ordinateur sur lequel vous développerez votre application pendant l'examen aura la dernière version du système de développement complet ou professionnel de LabVIEW. Contactez votre surveillant ou votre centre de test avant l'examen pour obtenir des détails supplémentaires et vous familiariser avec la version de LabVIEW que vous utiliserez pour développer votre application.
Reportez-vous à `http://www.ni.com/labview/buy/f` pour en savoir plus sur les fonctionnalités disponibles dans le système de développement complet ou professionnel de LabVIEW.

Notez qu'il ne vous sera pas donné plus de temps pour compenser un manque de connaissance de l'environnement LabVIEW. Si vous avez besoin d'un peu de temps pour personnaliser l'environnement, arrangez-vous avec le surveillant pour qu'il attende que vous soyez prêt à prendre l'examen avant de vous donner le kit de l'examen.

L'utilisation des ressources disponibles dans LabVIEW, comme l'*Aide LabVIEW*, les exemples et les modèles est autorisée pendant l'examen. Il est interdit d'utiliser des ressources ou VIs développés hors du cadre de l'examen.

L'examen CLA consiste en un projet fort similaire à celui sur lequel vous avez travaillé au cours de l'examen CLD.

Votre solution doit être transférée sur une clé USB qui doit être rendue au surveillant.

Remarque **Vous *ne devez* ni détacher l'agrafe, ni copier, ni reproduire, ni garder aucune section du document de l'examen ou de la solution. Le non respect de cette règle vous fera échouer à l'examen.**

Rubriques de l'examen CLA

1. Spécifications du projet
2. Organisation et hiérarchie du projet
3. Architecture et conception du projet
4. Conception et développement en équipe, et pratiques de standardisation
5. Outils réutilisables / Conception de composants

Remarque L'examen CLA est cumulatif et inclut les rubriques des examens CLAD et CLD.

Rubrique	Sous-rubrique
1. Spécifications du projet	a. Spécifications techniques b. Suivi des spécifications c. Développement de l'interface utilisateur d. Interface matérielle e. Gestion des erreurs
2. Organisation et hiérarchie du projet	a. Hiérarchie du projet LabVIEW b. Hiérarchie sur disque c. Chemins LabVIEW d. Hiérarchie modulaire
3. Architecture et conception du projet	a. Architecture du VI principal b. Architecture des modules / sous-VIs c. Architecture de communication par messages d. Module d'erreur e. E/S sur fichiers et base de données f. Architecture de simulation g. Conception d'interface utilisateur h. Méthodes de conception avancées i. Documentation des spécifications
4. Conception et développement en équipe, et pratiques de standardisation	a. Pratiques de développement LabVIEW b. Fonctionnalités modulaires c. Documentation pour l'achèvement par les développeurs d. API modulaires claires
5. Outils réutilisables / Conception de composants	a. Technologies LabVIEW b. Conception d'APIs c. Modèles de conception

Détails sur les rubriques de l'examen CLA

1. Spécifications du projet

a. Spécifications techniques

Déterminer et répertorier les spécifications suivantes à partir des spécifications du projet :

1. Spécifications de l'application : but et objectif de l'application
2. Spécifications de l'interface utilisateur : présentation et comportement des commandes avec lesquelles l'utilisateur agit
3. Spécifications fonctionnelles : la fonctionnalité des composants et leur interaction avec le système
4. Spécifications de cadencement : matériel ou logiciel, débordement de données événementielles, heure d'été
5. Spécifications de la gestion d'erreurs : mise en gardes, erreurs, erreurs critiques, séquence d'arrêt
6. Spécifications de matériel ou de simulation : éléments d'interface et l'exécution requis pour passer à des périphériques réels.
7. Spécifications des entrées et sorties : console, bases de données
8. Spécifications de l'initialisation et de l'arrêt : comportement de l'interface utilisateur et du programme au démarrage, à l'arrêt et en cas d'erreurs
9. Spécifications non fonctionnelles : précision, performances, capacité de modification
10. Suppositions et contraintes
    a) Une supposition fonctionnelle est un problème qui n'est pas clair dans les spécifications
    b) Une contrainte fonctionnelle est une décision de conception imposée par les spécifications

b. Suivi des spécifications

1. Identifier et satisfaire les spécifications
   a) Déterminer le niveau de détail des spécifications
   b) Placer les balises de spécifications dans l'architecture uniquement où les spécifications sont satisfaites
2. Méthodes ou (utilisation d') outils logiciels pour suivre les spécifications
   a) Utiliser le format spécifié pour les balises de spécifications pour le suivi à l'aide de Requirements Gateway.
   b) Utiliser le fichier de balises fourni

c. Développement de la GUI

1. Construire une interface utilisateur basée sur les spécifications
   a) Déterminer le type de commandes approprié en fonction des spécifications fonctionnelles
   b) Utiliser des définitions de type

2. Concevoir une interface qui répond aux spécifications fonctionnelles
   a) Utiliser les Règles de développement LabVIEW
   b) Organiser, modulariser, ou regrouper les composants de l'interface utilisateur pour suivre un processus ou une séquence logique
   c) Utiliser des techniques de développement LabVIEW avancées

d. Interface matérielle
   1. Utiliser l'abstraction pour séparer les modules de simulation des modules matériels
      a) Développer une API qui servira d'inferface avec le module du contrôleur
      b) Concevoir une interface extensible qui facilite la transition de la simulation au matériel
      c) Développer une méthode pour sélectionner les modules de simulation ou matériels
   2. Développer une architecture de simulation basée sur le matériel
      a) Sélectionner une architecture modulaire qui simule le matériel externe
      b) Sélectionner des composants d'interface utilisateur qui imitent la fonction du matériel

e. Gestion des erreurs
   1. Développer une gestion d'erreurs centralisée
      a) Démontrer des méthodes pour gérer les mises en garde, les erreurs critiques et les erreurs d'arrêt comme le définissent les spécifications
      b) Développer une architecture qui intègre le module d'erreur dans le VI principal et dans les autres modules
   2. Concevoir une méthode d'arrêt abstraite des modules de fonctionnalité pour répondre au module d'erreur

2. Organisation et hiérarchie du projet

a. Hiérarchie du projet LabVIEW
   1. Développer une hiérarchie de projet LabVIEW pour un développement en équipe
      a) Modules et leur hiérarchie
      b) Sous-VIs partagés, commandes personnalisées
      c) VIs de plug-in
      d) Bibliothèques de projet LabVIEW
      e) Fichiers de support (fichiers de documentation, de configuration et journaux)
   2. Utiliser une convention pour les noms

b. Hiérarchie sur disque
   1. Reproduire la hiérarchie du projet sur le disque
   2. Utiliser des dossiers remplis automatiquement
   3. Organiser la hiérarchie du projet et du disque par module ou d'après un autre schéma basé sur les fonctionnalités

c. Chemins
   1. Utiliser des chemins relatifs et demander au développeur d'en faire autant
d. Hiérarchie modulaire
   1. Organiser par module ou d'après un autre schéma basé sur les fonctionnalités
   2. Sous-dossiers basés sur les éléments du code comme, par exemple, les commandes et les sous-VIs des modules

3. Architecture et conception du projet

a. Architecture du VI principal
   1. Sélectionner une architecture modulaire extensible avancée qui permet :
      a) La gestion des événements de l'interface utilisateur et des événements utilisateur
      b) Le traitement asynchrone et parallèle des événements
      c) L'initialisation, l'arrêt, la persistance d'état et la restauration
      d) La gestion efficace des erreurs (de logique et d'exécution)
      e) Cadencement (basé sur événements ou des interrogations)
      f) Développement en équipe de modules fonctionnels
   2. Développer des structures de communication par messages de données et d'événements
   3. Développer une architecture pour gérer les données de configuration
   4. Développer des interfaces pour la simulation et les autres modules
   5. Utiliser les Règles de développement LabVIEW pour optimiser la mémoire
b. Architecture des modules / sous-VIs
   1. Sélectionner une architecture et un modèle de conception cohérents pour les modules et les sous-VIs
   2. Définir et développer une API claire
   3. Définir des connecteurs et des icônes cohérents
c. Architecture de communication par messages
   1. Modulariser un schéma de communication par messages pour l'initialisation, l'utilisation et l'arrêt
   2. Démontrer une méthode de communication par messages que les développeurs devront achever
   3. Démontrer un couplage faible du module de communication par messages
d. Module d'erreur
   1. Modulariser la gestion d'erreurs centralisée pour une initialisation, une utilisation et un arrêt clairs.
   2. Démontrer l'intégration de la gestion d'erreurs avec les modules de fonctionnalités.
   3. Gérer l'arrêt comme spécifié
   4. Démontrer l'enregistrement des données dans des fichiers journaux
   5. Établir des actions différentes selon l'importance de l'erreur

e. E/S sur fichiers et base de données
   1. Modulariser les E/S pour une initialisation, une utilisation et un arrêt clairs.
   2. Communiquer aux développeurs les méthodes d'accès qu'ils doivent implémenter
   3. Spécifier les formats de données et la conversion aux structures de données de l'application
   4. Gérer les données de configuration et l'enregistrement des erreurs dans des fichiers journaux
f. Architecture du module de simulation
   1. Sélectionner une architecture modulaire qui simule le matériel externe
   2. Concevoir une interface extensible qui facilite la transition de la simulation au matériel
   3. Sélectionner des composants d'interface utilisateur qui imitent la fonction du matériel
g. Conception d'interface utilisateur
   1. Utiliser les *Règles de développement LabVIEW*
   2. Organiser, modulariser, ou regrouper les composants de l'interface utilisateur pour suivre un processus ou une séquence logique
   3. Utiliser des techniques de développement LabVIEW avancées
h. Méthodes de conception avancées
   1. Développer une architecture pour une application modulaire, extensible, et facile à maintenir
   2. Implémenter, développer et améliorer les modèles de conception standard pour répondre aux besoins du projet
   3. Utiliser une conception événementielle pour les événements de l'interface utilisateur et définir des événements générés par l'utilisateur pour le cadencement, les erreurs, la signalisation, etc.
   4. Abstraire les fonctionnalités et développer une API claire et cohérente pour les modules et les sous-VIs
   5. Utiliser et normaliser des types de donnés et des structures de données extensibles
   6. Utiliser les techniques de conception orientée objet, de récursion, de VI Serveur et d'E/S sur fichier avancées
i. Documentation des spécifications
   1. Utiliser les *Règles de développement LabVIEW*
   2. Documenter les éléments suivants :
      a) L'architecture principale pour l'intégration des modules
      b) Les structures de données et le mécanisme de communication des données et des messages
      c) Modules, sous-VIs et interfaces (API)
      d) Module de simulation, interfaces et éléments requis pour passer de la simulation au module matériel

4. Conception et développement en équipe, et pratiques de standardisation

a. Pratiques de développement LabVIEW
   1. Établir et utiliser un style de développement cohérent : utiliser les *Règles de développement LabVIEW* ainsi que les normes développées par la compagnie
   2. Utiliser des modèles comme point de départ du développement
   3. Documenter les propriétés du VI, le diagramme et l'interface utilisateur (info-bulles, etc.)
   4. Développer des modules et des outils réutilisables pour uniformiser le développement

b. Architecture pour un développement modulaire
   1. Sélectionner une architecture et un modèle de conception cohérents pour les modules et les sous-VIs
   2. Définir des connecteurs et des icônes cohérents
   3. Définir une gestion d'erreurs et faire en sorte que les erreurs critiques soient traitées de manière appropriée
   4. Développer les principales structures et la communication par messages
   5. Développer avec suffisamment de détails pour que le développeur puisse implémenter les différentes spécifications

c. Instructions documentées pour que le développeur puisse achever l'application
   1. Documenter comment terminer le code des algorithmes, des transactions et de la logique pour qu'une équipe de développeurs puisse achever les fonctionnalités
   2. Documenter plusieurs étapes similaires en terminant une première étape en détaillant les spécifications, et en faisant référence à la technique utilisée dans cette première étape pour les étapes ultérieures
   3. Utiliser du code LabVIEW sur le diagramme pour illustrer la technique et compléter ceci avec de la documentation comportant des instructions pour les développeurs

d. API modulaires claires
   1. Définir les API des modules et des sous-VIs
   2. Développer des API pour les modules de fonctionnalité afin de permettre la modularité et l'abstraction
   3. Développer l'architecture avec des API pour la gestion des erreurs, l'initialisation et l'arrêt

5. Outils réutilisables / Conception de composants

a. Technologies LabVIEW
   1. Déterminer la méthode optimale pour développer un composant réutilisable ou un outil d'amélioration de la productivité à partir des technologies suivantes :
      a) Commandes personnalisées
      b) Fusion de VIs
      c) Sous-VI

   d) CommandeX
   e) VI modèle

b. Conception d'APIs
   1. Développer une API simplifiée pour encapsuler des fonctions LabVIEW avancées
   2. Développer des VIs gestionnaires pour traiter les tâches courantes comme, par exemple, la gestion des références aux files d'attente, les événements utilisateur, etc.
   3. Utiliser les options d'accès du projet pour interdire ou autoriser l'accès aux éléments des bibliothèques

c. Modèles de conception
   1. Sélectionner des modèles de conception appropriés pour les modules et les sous-VIs, en fonction des spécifications fonctionnelles
   2. Utiliser de la documentation pour décrire comment achever les éléments courants du modèle de conception.

Examen CLA

Dans l'examen CLA, vous devrez concevoir une architecture qui répond aux spécifications indiquées dans une spécification de projet.

Durée de l'examen : 4 heures
Style d'examen : développement d'architecture d'application
Seuil d'obtention : 70 %

Développement d'architecture d'application :
Vous devez développer un framework d'application constitué d'un VI principal, de modules, de sous-VIs de support et de commandes personnalisées (définitions de type). Un module est un sous-VI ou un groupe de sous-VI qui réalise un ensemble de fonctions. Un module peut avoir sa propre hiérarchie de sous-VIs.

Remarque Vous *n'avez pas besoin* de soumettre une application fonctionnelle. Les détails fonctionnels des éléments requis doivent être documentés dans le VI principal, les modules et les sous-VIs. Vous devez fournir cette documentation dans l'architecture pour permettre aux développeurs de votre équipe de développer les fonctionnalités.

L'architecture a les spécifications minimales suivantes :

a. Développer une hiérarchie de projet
b. Développer un VI principal. Le VI principal doit comporter les éléments suivants :
   i. Interface utilisateur modulaire
   ii. Architecture pilote
   iii. Principales structures de données
   iv. Méthode(s) de communication des événements, des données, du cadencement et des erreurs
   v. Gestion des erreurs
   vi. Modules et/ou sous-VIs complètement connectés
c. Développer des squelettes (stubs) de modules et sous-VIs, qui *ne doivent pas* inclure de logique fonctionnelle détaillée, mais doivent inclure :
   i. Les entrées, les sorties, l'icône et le connecteur
   ii. L'architecture et l'API
   iii. Les principales structures de données internes
   iv. La gestion et la communication des erreurs
   v. Des instructions ou des commentaires répertoriant les fonctionnalités avec suffisamment de détails pour qu'un développeur puisse achever les fonctionnalités du VI
d. Développer une interface pour la simulation de matériel dans un module distinct ou au sein du VI principal ou d'un autre module, en fonction de votre conception.
e. Développer un mécanisme de communication entre les processus
f. Développer une stratégie de gestion des erreurs et d'arrêt

Suivi des spécifications
Les spécifications du projet décriront en détail les spécifications requises identifiées par un identificateur unique. Pour prouver qu'une spécification a été satisfaite, vous devez inclure l'ID de la spécification dans la documentation de votre architecture. Les spécifications peuvent être satisfaites dans toute partie de la documentation de l'architecture, y compris :

- Propriété de documentation d'un VI
- Propriété de documentation d'une commande
- Propriété de documentation du projet ou d'une bibliothèque
- Commentaires sur la face-avant ou le diagramme

Une spécification particulière peut être satisfaite par plusieurs sections de code si toutes ces sections sont nécessaires pour la satisfaire.
Pour satisfaire une spécification, le texte suivant doit se trouver dans la documentation du code : `[Covers: ID]` Par exemple : `[Covers: CD1]`

La clé USB fournie comporte un fichier texte qui contient toutes les balises. Ce fichier est fourni pour vous permettre de placer plus facilement les balises dans le code de votre application.

Remarque Un outil de suivi des spécifications (Requirements Gateway) sera utilisé pour vérifier quelles spécifications ont été satisfaites ; il faut donc suivre exactement la syntaxe précédente.

Reportez-vous aux exemples d'examen CLA pour voir comment documenter les spécifications remplies dans le VI, ainsi que les instructions ou commentaires qu'il faut inclure dans le VI pour qu'un développeur puisse terminer l'implémentation.

Évaluation :
L'allocation des points dans l'examen CLA est la suivante : (total de 100 points)

- Style de l'interface utilisateur et du diagramme : 10 points
- Documentation : 20 points
- Couverture des spécifications : 30 points
- Développement de l'architecture : 40 points

Ressources pour la préparation à l'examen CLA

Utilisez les ressources supplémentaires suivantes pour vous préparer à l'examen :

- Managing Software Engineering in LabVIEW
- Advanced Architectures for LabVIEW
    - Formation dispensée par un instructeur
    - Auto-formation en utilisant les manuels du cours
- Exemples d'examen CLA :
    - www.ni.com/claprep

Scénarios des examens CLA

Le tableau suivant répertorie les scénarios d'examen que vous pourriez recevoir pour développer une solution pour votre examen CLA. Cette liste a pour but de vous donner une idée des examens administrés ; cependant, des variations sont possibles pour chaque examen.

Scénario d'examen	Description
Station de lavage automatique	La station de lavage automatique simule la sélection d'options de lavage personnalisées et la conduite d'une voiture d'un bout à l'autre de la station.
Machine à café	La machine à café simule le stockage des ingrédients et effectue des opérations de mouture, de percolation et de distribution pour préparer de l'eau chaude, du café et du café au lait.
Machine à pizzas	La machine à pizzas simule la création d'une recette de pizza personnalisée et la préparation, la cuisson et la coupe de la pizza.
Système de sécurité	Le système de sécurité multizone simule les fonctions d'activation, de désactivation de toutes ou certaines des zones, d'anti-sabotage et de déclenchement de l'alarme.
Thermostat	Le thermostat simule le contrôle programmable du chauffage ou du refroidissement par un système de chauffage, de ventilation et de conditionnement (CVC).