Présentation de la certification

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

1. Examen CLAD (Certified LabVIEW Associate Developer)
2. Examen CLD (Certified LabVIEW Developer)
3. Examen CLA (Certified LabVIEW Architect)

Chaque niveau de certification est prérequis pour passer au niveau suivant. 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 aux exigences de chaque examen.

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 avez la capacité 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 échelle. Le niveau typique pour un professionnel qui passe le CLD est une expérience cumulée de 12 à 18 mois environ dans le développement d'applications LabVIEW de moyenne ou grande échelle.

L'obtention du CLA démontre la maîtrise de la création d'architectures pour des applications LabVIEW dans un environnement à plusieurs développeurs. L'obtention de cet examen montre que non seulement vous possédez l'expertise technique et l'expertise de développement de logiciel nécessaires pour diviser une spécification de projet en composants LabVIEW gérables, mais aussi que vous avez la capacité de mener à bien un projet en utilisant de manière efficace les outils de gestion de configuration et de projets. 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 échelle.

Présentation de l'examen

La version 2011 du Système de développement complet de LabVIEW ou une version ultérieure doit être installée sur l'ordinateur de test pour développer votre application. Passé le 1er octobre 2012, LabVIEW 2012 devra être installé.
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 allez utiliser pour développer votre application.
Vous pouvez demander au surveillant de vous donner quelques minutes avant l'examen afin de personnaliser l'environnement LabVIEW en fonction de vos besoins et de vous familiariser avec l'environnement. Le surveillant ne distribuera l'examen que lorsque vous serez prêt à travailler sur l'examen.
Notez qu'il ne vous sera pas donné plus de temps pour compenser un manque de connaissance de l'environnement LabVIEW.
Reportez-vous à la comparaison des Systèmes de développement LabVIEW pour en savoir plus sur les fonctionnalités du Système de développement complet de LabVIEW.

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

L'examen valide la maîtrise de la résolution de problèmes, les connaissances et l'expérience du développement d'applications de mesure et d'automation avec LabVIEW. L'examen ne comporte que le développement logiciel et ne nécessite pas de matériel.

L'utilisation des ressources disponibles dans LabVIEW, comme l'*Aide LabVIEW*, les exemples et les modèles est autorisée pendant l'examen. Les ressources et VIs développés hors du cadre de l'examen sont interdits.

Le paquet de l'examen contiendra une clé USB comportant un VI et des commandes dans une hiérarchie de dossiers que vous devez utiliser pour développer votre application. Vous ne devez ni renommer le VI principal ou les éléments fournis, ni modifier la hiérarchie de dossiers. Tous vos VIs de développement et toutes vos commandes doivent être enregistrés dans la hiérarchie de dossiers fournie.

Des spécifications détaillées vous seront fournies pour l'application. Les spécifications comprennent des exigences générales et techniques pour l'application. Vous **ne devez** ni détacher l'agrafe, ni copier ou reproduire une section du document de l'examen. Le non respect de cette règle entraînera l'annulation de votre examen.

Lorsque vous avez terminé l'examen, vous devez transférer la solution sur la clé USB fournie. Veuillez valider la solution copiée sur la clé USB avant de la rendre au surveillant.

Rubriques de l'examen

1. Concepts de conception
2. Conception de l'interface utilisateur
3. Style et mise en page du diagramme
4. Pratiques de programmation
5. Pratiques de conception des sous-VIs
6. Choix de l'architecture
7. Cadencement
8. Gestion des erreurs
9. Documentation
10. Test

Rubriques de l'examen (présentation)

Rubrique	Sous-rubrique
1. Concepts de conception	a. Modularité, évolutivité, lisibilité et maintenance facile b. Cohésion et indépendance c. Conception hiérarchique d. Structure de fichiers
2. Pratiques de conception de l'interface utilisateur (fenêtre de la face-avant)	a. Choix des couleurs b. Groupage et alignement d'objets c. Définition des propriétés d. Personnalisation des objets e. Gestion d'états i. Statique ou dynamique ii. Lors de l'initialisation et de l'arrêt de l'application f. Style d'icônes
3. Pratiques de conception du diagramme	a. Flux de données b. Amélioration de la lisibilité
4. Pratiques de programmation	a. Éléments de données b. Fonctions et sous-VIs c. Structures de programmation d. Structures de données e. Références, nœuds de propriété
5. Pratiques de conception des sous-VIs	a. Modularité et cohésion b. Mise en page du diagramme c. Connecteur et icône
6. Sélection du modèle de conception	a. Évolutivité et maintenance facile b. Réactif et fluide c. Modèles de conception : i. Machine à états simple ii. Gestionnaire d'événements de l'interface utilisateur iii. Gestionnaire de messages dans une file d'attente iv. Producteur/consommateur (données) v. Producteur/consommateur (événements) vi. Variable globale fonctionnelle
7. Cadencement	a. Fonctions de cadencement b. Mécanismes de cadencement i. Timeout de la structure Événement ii. Timeout de la fonction de synchronisation iii. Structures cadencées c. Variables globales fonctionnelles et VIs Express de cadencement
8. Erreurs	a. Gestion des erreurs b. Rapport d'erreurs
9. Documentation	a. Fenêtre de la face-avant b. Diagramme c. Propriétés du VI
10. Test	a. Code et révision de la documentation b. Fonctionnalité d. Erreurs

Informations détaillées sur les rubriques du CLD

1. **Concepts de conception**
   a. Modularité, évolutivité, lisibilité et maintenance facile
      1. Développer une application LabVIEW (VI) qui soit :
         a) Modulaire – Les fonctionnalités du VI sont sous-divisées en modules ou sous-VIs.
         b) Flexible – Le VI nécessite peu ou pas de changement dans l'interface utilisateur ou le diagramme pour gérer des ensembles de données plus grands ou des états de programmation supplémentaires.
         c) Lisible – Le VI donne des informations sur ses fonctionnalités grâce à une bonne utilisation de la documentation et un bon style de programmation.
         d) Facile à maintenir – Le VI permet d'effectuer des modifications sans changer le but initial du module ou de l'application.
   b. Cohésion et indépendance
      1. Développer des modules LabVIEW qui :
         a) Aient une bonne cohésion – Le module a un but clairement défini et publié.
         b) Soient indépendants – Le module dépend le moins possible d'autres modules pour effectuer ou complémenter sa fonctionnalité.
   c. Conception hiérarchique
      1. Développer un VI LabVIEW qui utilise les techniques précédentes pour créer une conception hiérarchique logique
   d. Structure de fichiers
      1. Organiser les VIs dans le système de fichiers pour refléter la nature hiérarchique du logiciel
      2. Créer un dossier pour l'application et lui donner un nom pertinent
      3. Rendre le VI (de niveau principal) accessible dans le dossier de l'application
      4. Créer des dossiers séparés pour les sous-VIs et les commandes

2. **Pratiques de conception de l'interface utilisateur (face-avant)**
   a. Choix des couleurs
      1. Concevoir l'interface utilisateur d'un VI en étant cohérent dans le choix de couleurs du système
      2. Utiliser des couleurs pastel quand cela est nécessaire et éviter d'utiliser des couleurs vives
      3. Utiliser les consignes de la rubrique *LabVIEW Style Checklist* de l'*Aide LabVIEW* pour choisir les couleurs des arrières-plans et des objets de l'interface utilisateur
   b. Groupage et alignement d'objets
      1. Grouper les objets de l'interface utilisateur qui ont un lien logique en utilisant des tableaux, des clusters ou des décorations
      2. Aligner les objets et leurs étiquettes pour offrir une mise en page cohérente et uniforme

c. Définition des propriétés
   1. Choisir les paramètres appropriés pour l'objet de la face-avant pour faciliter son utilisation et améliorer ses performances
d. Personnalisation des objets
   1. Changer l'apparence esthétique d'un objet de la fenêtre de la face-avant
   2. Étendre l'évolutivité de l'application en créant une définition de type ou définition de type stricte pour une commande personnalisée
e. Gestion d'états
   1. Définir la valeur ou les attributs d'une commande en utilisant de façon statique la boîte de dialogue de propriétés d'un objet ou en utilisant de façon dynamique des nœuds de propriété
   2. Initialiser ou définir les valeurs d'une commande au chargement, au démarrage et à l'arrêt de l'application
f. Style d'icônes
   1. Concevoir des icônes de VIs de niveau principal et de sous-VIs qui représentent la fonctionnalité du module ou de l'application
   2. Conserver un style d'icône uniforme et cohérent entre le VI principal et les sous-VIs

3. Pratiques de conception du diagramme

a. Flux de données
   1. Forcer le flux de données désiré en utilisant des terminaux d'erreur sur les VIs et les nœuds de propriété
   2. Forcer le flux de données désiré en utilisant des structures Séquence pour les VIs et les fonctions qui n'ont pas de terminaux d'erreur
b. Amélioration de la lisibilité
   1. Développer les diagrammes pour qu'ils soient visibles avec une résolution d'écran de 1024 x 768
   2. Limiter la taille du diagramme de façon à ce que l'utilisateur n'ait à le faire dérouler que dans une direction.
   3. Espacer les VIs et les fonctions régulièrement, éviter de mettre trop de VIs ou de fonctions dans une petite zone
   4. Aligner les VIs et les fonctions à intervalles réguliers en utilisant une disposition cohérente
   5. Dans la mesure du possible, éviter les coudes dans les fils de liaison
   6. Connecter les fils de liaison pour qu'ils soient visiblement connectés aux bons terminaux
   7. Câbler les VIs et les fonctions pour qu'ils suivent le flux de données de gauche à droite et de haut en bas
   8. Éviter de cacher les fils de liaison sous des structures ou des bords de structures
   9. Éviter de faire chevaucher des tunnels sur les bords des structures
   10. Éviter d'utiliser des couleurs pour faire la distinction entre des sections du diagramme, utiliser les couleurs du système si besoin est

4. Pratiques de programmation

a. Éléments de données
   1. Utiliser les types de données appropriés pour les commandes, indicateurs et constantes
   2. Lire directement les commandes, éviter d'utiliser des variables locales ou globales ou des nœuds de propriété
   3. Mettre les indicateurs à jour directement, éviter d'utiliser des variables locales ou globales ou des nœuds de propriété
   4. Utiliser des variables locales pour mettre les commandes à jour
   5. Utiliser des nœuds de propriété pour définir les attributs des commandes et indicateurs
   6. N'utiliser de références que si des commandes ou indicateurs de la face-avant sont affectés par un sous-VI
   7. Protéger l'accès aux variables locales et globales pour éviter des situations de compétition
   8. Utiliser des éléments de données à définition de type pour garder le même type pour toutes les instances et améliorer l'évolutivité
   9. Utiliser des constantes pour les éléments de données qui n'ont pas besoin d'être accessibles à partir de l'interface utilisateur

b. Fonctions et sous-VIs
   1. Optimiser l'utilisation des fonction, utiliser un minimum de fonctions pour effectuer une tâche
   2. Éviter la coercition des données aux entrées
   3. Utiliser les terminaux d'erreur quand ils sont disponibles
   4. Fermer les références qui ont été ouvertes de manière explicite

c. Structures de programmation
   1. Sélectionner une structure de programmation appropriée pour le VI
   2. Initialiser les registres à décalage là où cela est nécessaire et identifier les conséquences des registres à décalage non initialisés
   3. Éviter d'utiliser des structures profondément imbriquées, limiter l'imbrication à deux niveaux
   4. Utiliser une structure Séquence déroulée à une étape pour forcer le flux de données désiré là où un fil de liaison n'est pas disponible
   5. Éviter d'utiliser des variables locales de séquence pour transférer des données ou des informations d'état
   6. Dans la mesure du possible, remplacer les structures Séquence par des structures Condition pour améliorer l'évolutivité
   7. Éviter d'utiliser des tunnels par défaut sur le cadre des structures
   8. Utiliser une structure Événement pour gérer les événements de l'interface utilisateur
   9. Utiliser une structure cadencée pour gérer les événements de cadencement de manière déterministe

d. Structures de données
   1. Grouper les éléments de données associés de façon logique dans des structures de données appropriées
   2. Créer des structures de données fonctionnelles sur le diagramme pour grouper et gérer des données et des informations d'état qui n'ont pas besoin d'être accessibles à partir de l'interface utilisateur

e. Références, nœuds de propriété (obtention, fermeture de références)
   1. Utiliser des nœuds de propriété liés implicitement pour affecter les attributs d'objets dans un même VI
   2. N'utiliser de références à des objets de la face-avant que s'ils sont affectés par un sous-VI
   3. Identifier l'impact de l'utilisation des nœuds de propriété sur les performances et appliquer ces nœuds de façon appropriée
   4. Fermer les références qui ont été ouvertes de manière explicite

5. Pratiques de conception des sous-VIs

a. Cohésion et modularité
   1. Concevoir un sous-VI de façon à ce qu'il effectue une fonction clairement définie
   2. Appeler des fonctions et des sous-VIs pour aider le sous-VI à effectuer sa tâche

b. Fenêtres de la face-avant et du diagramme
   1. Concevoir l'interface utilisateur d'un sous-VI pour que les sections d'entrée, de sortie et de données soient clairement définies
   2. Inclure des clusters d'entrée et de sortie d'erreur et les câbler au connecteur
   3. Câbler le terminal entrée d'erreur au terminal de sélection d'une structure Condition Erreur / Pas d'erreur
   4. S'assurer que la condition Erreur transfère l'erreur en amont
   5. Placer tout le code des sous-VIs dans une condition Pas d'erreur et s'assurer que le fil de liaison d'erreur est connecté aux terminaux d'erreur des fonctions, sous-VIs et nœuds de propriété
   6. Fusionner les erreurs de différentes sources et transmettre l'erreur au VI appelant via le terminal sortie d'erreur
   7. Utiliser les pratiques de programmation de la section *Pratiques de programmation*

c. Connecteur et icône
   1. Utiliser les techniques de la rubrique *LabVIEW Style Checklist* de l'*Aide LabVIEW* pour développer un connecteur
   2. Identifier quels terminaux sont nécessaires, recommandés ou optionnels
   3. Créer une icône pour le VI principal et les sous-VIs qui identifie leur fonctionnalité
   4. Conserver un style d'icône cohérent entre le VI principal et les sous-VIs

6. Sélection du modèle de conception

a. Évolutivité et maintenance facile
   1. Identifier le modèle de conception le plus approprié pour le VI principal en analysant les spécifications du projet
   2. Identifier le modèle de conception le plus approprié pour les sous-VIs afin d'obtenir la fonctionnalité désirée
   3. Utiliser une architecture qui ne limite pas l'évolutivité ou la maintenance
   4. Utiliser des structures de données à définition de type pour la gestion des données et des états

b. Réactif et fluide
   1. Utiliser un modèle de conception qui permet d'interagir avec l'interface utilisateur dans un délai de 100 ms et qui met à jour les indicateurs à une fréquence raisonnable
   2. Utiliser un modèle de conception et des techniques qui n'utilisent pas 100 % des ressources du processeur

c. Modèles de conception
   1. Sélectionner un modèle de conception de machine à états simple pour le VI de niveau principal dans lequel l'IU est interrogée
   2. Sélectionner un modèle de conception de machine à états simple dans un sous-VI qui doit exécuter une ou plusieurs fonctions ou sous-VIs en fonction d'un état en entrée
   3. Sélectionner un modèle de conception de gestionnaire d'événements d'interface utilisateur dans lequel une réponse rapide est nécessaire quand l'interface utilisateur est utilisée
   4. Sélectionner un modèle de conception de gestionnaire de messages dans une file d'attente pour stocker un ou plusieurs états
   5. Sélectionner un modèle de conception producteur/consommateur (données) pour générer et stocker des données ou des états dans la boucle productrice en parallèle avec d'autres boucles consommatrices de données
   6. Sélectionner un modèle de conception producteur/consommateur (événements) pour répondre aux événements de l'interface utilisateur dans la boucle productrice et déléguer le traitement de l'événement à une ou plusieurs boucles consommatrices
   7. Sélectionner un modèle de conception de variable globale fonctionnelle dans un sous-VI au lieu d'une variable globale pour améliorer les performances et le contrôle d'accès

7. Cadencement

a. Exécution et cadencement logiciel
   1. Déterminer la fonction de cadencement appropriée pour contrôler la vitesse à laquelle le VI s'exécute sur le système et permettre au processeur de répondre aux événements externes et aux tâches du système
   2. Déterminer le mécanisme le mieux adapté à l'exécution d'une tâche ou d'une opération logicielle pendant ou après un laps de temps défini

b. Fonctions de cadencement
   1. Utiliser la fonction Attendre pour contrôler la vitesse d'exécution d'une boucle et permettre au processeur de répondre aux tâches du système et aux événements externes
   2. Utiliser la fonction Attendre un multiple de ms pour contrôler la vitesse d'exécution de la boucle et synchroniser plusieurs boucles
   3. Éviter d'utiliser des fonctions d'attente pour cadencer une opération logicielle
   4. Utiliser les fonctions Compteur d'impulsions d'horloge et Date et heure en secondes pour cadencer des opérations logicielles
   5. Prendre en compte le fait que la fonction Compteur d'impulsions d'horloge peut repasser à zéro
   6. Avertir l'utilisateur en cas d'erreurs dues à des changements dans l'horloge système si la fonction Date et heure en secondes est utilisée

c. Mécanismes de cadencement
   1. Utiliser le timeout sur la structure Événement pour cadencer des opérations logicielles
   2. Utiliser l'entrée timeout des fonctions de file d'attente et de notification pour cadencer des opérations logicielles
   3. Utiliser une structure de cadencement pour effectuer des opérations de cadencement logiciel déterministes

d. Variables globales fonctionnelles et VIs Express de cadencement
   1. Utiliser le VI Express Temps écoulé pour cadencer des opérations logicielles
   2. Développer des sous-VIs de cadencement en utilisant le modèle de conception de variable globale fonctionnelle et soit la fonction Compteur d'impulsions d'horloge, soit la fonction Date et heure en secondes

8. Erreurs

a. Gestion des erreurs
   1. Initialiser le cluster d'erreur au démarrage de l'application
   2. Câbler les terminaux d'erreurs des fonctions, sous-VIs et nœuds de propriété
   3. Fusionner les erreurs de plusieurs sources quand c'est nécessaire
   4. Définir des codes d'erreur personnalisés en utilisant le VI Gestionnaire d'erreur général
   5. Déterminer les actions à effectuer quand une erreur a lieu
   6. Utiliser la gestion d'erreur corrective quand c'est possible
   7. S'assurer que toutes les boucles en cours d'exécution s'arrêtent en cas d'erreur critique en utilisant directement le fil de liaison d'erreur ou une condition de gestion d'erreur

b. Rapport d'erreurs
   1. Utiliser les VIs de gestion d'erreur ou de boîte de dialogue pour rapporter les erreurs ou les mises en garde

9. Documentation

a. Interface utilisateur (face-avant)
   1. Étiqueter les objets de l'interface utilisateur avec des noms représentatifs de leur fonction
   2. Inclure des info-bulles concises pour les commandes de l'interface utilisateur
   3. Donner des instructions d'utilisation spéciales quand cela est nécessaire
   4. Donner un nom de groupe approprié aux commandes groupées dans des clusters ou des décorations

b. Diagramme
   1. Fournir de la documentation concise sur l'algorithme général du VI principal et des sous-VIs
   2. Documenter chaque structure de programmation avec une brève description de ses fonctionnalités
   3. Étiqueter les fils de liaison pour indiquer les données transmises par le fil et la direction du flux de données
   4. Étiqueter les constantes avec des noms représentatifs de leur fonction
   5. Utiliser les pratiques de programmation d'auto-documentation

c. Propriétés du VI
   1. Fournir de la documentation concise dans les propriétés du VI principal en indiquant sa fonctionnalité générale
   2. Fournir de la documentation concise dans les propriétés du VI des sous-VIs en indiquant la fonctionnalité générale du sous-VI, sa description et le type de données des terminaux

10. Test

a. Code et révision de la documentation
   1. Revoir l'interface utilisateur, le diagramme et la conception du programme et s'assurer que ces éléments correspondent aux exigences précédentes et à la rubrique *LabVIEW Style Checklist* de l'*Aide LabVIEW*
   2. Revoir la documentation et s'assurer qu'elle correspond aux exigences précédentes et à la rubrique *LabVIEW Style Checklist* de l'*Aide LabVIEW*

b. Fonctionnalité
   1. S'assurer que tous les sous-VIs sont liés au VI principal quand celui-ci est ouvert et qu'aucune flèche d'exécution brisée n'apparaît
   2. S'assurer que des chemins relatifs sont utilisés pour accéder aux fichiers de données de l'application
   3. S'assurer que chaque sous-VI produit le résultat souhaité en exécutant le VI avec des données de test
   4. Tester l'application intégrée pour s'assurer qu'elle effectue bien la fonctionnalité spécifiée
   5. S'assurer que l'application n'utilise pas 100 % des ressources du processeur du système et qu'elle répond aux interactions avec l'interface utilisateur dans un délai de 100 ms
   6. Tester les commandes et indicateurs de la face-avant pour vérifier que leur paramétrage est correct à l'initialisation et à l'arrêt de l'application

c. Erreurs
   1. Tester l'application pour voir si elle produit une erreur pour une entrée de données inattendue dans les commandes de l'interface utilisateur
   2. Tester si les erreurs sont gérées correctement dans les sous-VIs et transmises aux appelants respectifs
   3. Tester si les erreurs sont gérées correctement et rapportées à l'utilisateur

Critères d'évaluation de l'examen CLD

L'examen CLD contient un total de 40 points alloués de la façon suivante :

- Style de programmation : **15 points**
- Fonctionnalité : **15 points**
- Documentation : **10 points**

  Seuil d'obtention : (70 %) **28 points**

Les critères suivants sont pris en considération pendant l'évaluation de l'examen :

Style

- Est-ce que l'application suit les directives de développement de LabVIEW ?
- Le VI est-il
  - Lisible ?
  - Construit pour être évolutif ?
  - Facile à maintenir ?
  - Trop complexe ?
- Le VI est-il construit de façon professionnelle ?
  - Le VI utilise-t-il des structure ou modèles de conception LabVIEW ?
  - L'exigence minimale est d'utiliser une machine à états.
  - Le VI est-il organisé de façon hiérarchique ?
  - Le code répété devrait se trouver dans des sous-VIs
  - Est-ce que les commandes énumération à définition de type sont utilisées pour définir des états ?
- Est-ce que le VI utilise des variables superflues ?
- Les types de données, gammes et format/précision sont-ils correctement utilisés pour les commandes de la face-avant ?
- Les données sont-elles groupées dans des structures de données appropriées : tableaux ou clusters ?
- Le VI utilise-t-il des structures imbriquées trop profondément (2 niveaux ou plus) ?
- Le VI utilise-t-il des structures Séquence pour autre chose que l'initialisation ou le nettoyage ?
- Le VI utilise-t-il des variables globales et locales ?
  - Des variables locales peuvent être utilisées pour mettre à jour des commandes.
  - Les variables globales sont-elles protégées pour éviter des situations de compétition ?
- Les nœuds de propriété (valeurs) sont-ils utilisés pour mettre à jour des indicateurs ?
- La face-avant et le diagramme sont-ils bien organisés ?
  - Les diagrammes contiennent-ils trop d'éléments dans des zones trop petites ?
- Y-a-t-il des coudes superflus dans les fils de liaison ?

- Est-ce que les objets et les fils de liaison se chevauchent ?
- Y-a-t-il des fils de liaison sous des structures ou des bords de structures ?
- Les terminaux d'erreur des VIs sont-ils câblés ?
- Les références sont-elles fermées correctement ?
- Le VI est-il optimisé pour la mémoire et les performances ?

Fonctionnalité

- La flèche **Exécuter** est-elle brisée ?
- Le VI remplit-il les exigences listées dans les spécifications ?
- La logique est-elle correcte pour les entrées et les sorties des booléens ?
- Le VI répond-il aux entrées de l'utilisateur dans la limite de temps spécifiée (100 millisecondes) ?
- Le VI ou sous-VI utilise-t-il 100 % des ressources du processeur ?
- Les E/S sur fichiers sont-elles implémentées correctement ?
- L'application s'arrête-t-elle en cas d'erreur ?

Documentation

- Le VI est-il bien documenté dans **Fichier»Propriétés du VI** ?
- Les sous-VIs sont-ils documentés ?
- Les fils de liaison sont-ils documentés avec les étiquettes appropriées ?
- La fonctionnalité est-elle documentée ?
  - Au niveau du diagramme.
  - Au niveau de la structure principale et de la structure imbriquée.
- Les commandes et indicateurs de la face-avant ont-ils des noms descriptifs ?
- Les VIs ont-ils des icônes descriptives ?
- Les constantes sont-elles documentées ?
- Les commandes de la face-avant ont-elle des info-bulles ?
- Le VI de niveau principal a-t-il une icône personnalisée ?
  - Tous les sous-VIs ont-ils un style d'icône cohérent ?

Ressources pour la préparation à l'examen CLD

Utilisez les ressources suivantes pour vous préparer à l'examen : Vous pouvez aussi utiliser les ressources données dans le guide de préparation du CLAD si vous avez besoin de rafraîchir vos connaissances sur certaines rubriques.

Préparation à l'examen CLD

- https://lumen.ni.com/nicif/f/ekitcldexmprp/content.xhtml (contient des liens vers les guides de préparation, les exemples d'examens et un webcast)

Cours de formation à votre rythme ou dispensés par un instructeur

- **http://sine.ni.com/tacs/app/overview/p/ap/of/lang/fr/pg/1/sn/n24:12725/id/1582/**
- **http://sine.ni.com/tacs/app/overview/p/ap/of/lang/fr/pg/1/sn/n24:12753/id/1583/**
- **http://sine.ni.com/tacs/app/overview/p/ap/of/lang/fr/pg/1/sn/n24:12754/id/1584/**
- **http://sine.ni.com/tacs/app/overview/p/ap/of/lang/fr/pg/1/sn/n24:12417/id/1588/** (ne fait pas partie des cours fondamentaux, mais est un excellent cours complémentaire pour l'examen)

Examens CLD

Le tableau suivant répertorie les scénarios d'examen que vous pourriez recevoir pour développer une solution pour votre examen CLD. 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
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.
Four à micro-ondes	Le four à micro-ondes simule une cuisson manuelle ou préconfigurée de recette à plusieurs étapes.
Système de sécurité	Le système de sécurité multizone simule les fonctions d'activation, de désactivation de toutes ou certaines des zones, de sabotage et de déclenchement de l'alarme.
Thermostat	Le thermostat simule le contrôle programmable du chauffage et du refroidissement par un système de chauffage, de ventilation et de conditionnement d'air.
Tapis de course	Le tapis de course simule un mode de programme manuel ou à plusieurs étapes qui contrôle la vitesse, la pente et le temps et qui fait le suivi de la durée de la course et de la distance parcourue.
Distributeur automatique	Le distributeur automatique simule le stockage, l'achat et la distribution de produits avec de la monnaie américaine.