Certification Overview

The National Instruments LabVIEW Certification Program consists of the following three certification levels:

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

Each level is a prerequisite for the next level of certification.

A CLAD demonstrates a broad and complete understanding of the core features and functionality available in the LabVIEW Full Development System and possesses the ability to apply that knowledge to develop, debug, and maintain small LabVIEW modules. The typical experience level of a CLAD is approximately 6 to 9 months in the use of the LabVIEW Full Development System.

A CLD demonstrates experience in developing, debugging, and deploying and maintaining medium to large scale LabVIEW applications. A CLD is a professional with an approximate cumulative experience of 12 to 18 months developing medium to large applications in LabVIEW.

A CLA demonstrates mastery in architecting LabVIEW applications for a multi-developer environment. A CLA not only possesses the technical expertise and software development experience to break a project specification into manageable LabVIEW components but has the experience to see the project through by effectively utilizing project and configuration management tools. A CLA is a professional with an approximate cumulative experience of 24 months in developing medium to large applications in LabVIEW.

Note The CLAD certification is a prerequisite to taking the CLD exam.
The CLD certification is a prerequisite to taking the CLA exam.
There are no exceptions to this requirement for each exam.

Exam Overview

Product: LabVIEW Full Development System version 8.0 for Windows. Refer to `ni.com/labview/how_to_buy.htm` for details on the features available in the LabVIEW Full Development System.

Exam Duration: 1 hour
Number of Questions: 40
Style of Questions: Multiple-choice
Passing grade: 70%

The exam validates application knowledge and not the ability to recall menu steps or names of VIs and components.

The use of LabVIEW or any other external resources is prohibited during the exam. For assistance and wherever appropriate, screenshots from the *LabVIEW Help* are provided in the exam.

To maintain the integrity of the exam, you may not copy or reproduce any section of the exam. Failure to comply will result in failure. In areas where the exam is deployed as a paper based exam, detaching the binding staple will result in failure without evaluation.

Exam Logistics

United States and Europe: The CLAD exam can be taken at Pearson Vue test centers. The exam is computer-based and results are available immediately upon completion of the exam. Refer to `www.pearsonvue.com` for more details and scheduling.

Asia: The exam is paper-based, for which the evaluations and results take about 4 weeks. Please contact your National Instruments local office for details and scheduling.

For general questions or comments, email: `certification@ni.com`.

Exam Topics

1. LabVIEW programming concepts
2. LabVIEW environment
3. Software constructs in LabVIEW
4. Programming VIs and functions
5. Data communication and synchronization VIs and functions
6. VI Server VIs and functions
7. Error handling VIs and functions
8. Design patterns
9. SubVI design
10. Debugging tools and techniques
11. VI design and documentation (style) practices
12. Memory, performance, and determinism

Exam Topics (Overview):

Topic	Sub Topic
1. LabVIEW programming principles	a. Data flow b. Polymorphism
2. LabVIEW environment	a. Front panel window, block diagram, and connector pane b. Menus and palettes c. Configuration options
3. Software constructs in LabVIEW	a. Front panel window and block diagram objects i. Controls, indicators, IO controls, and refnums ii. Terminals, constants, and nodes iii. Palettes, update modes, and legends of charts and graphs iv. Mechanical action of Boolean objects v. Property Nodes b. Data types and data structures i. Numeric, string, Boolean, and path data types ii. Array and cluster data types iii. Waveform and timestamp data types iv. Variant data types c. Working with objects and data types on front panel windows i. Ranges, formats, representation, and scaling ii. Customizing controls iii. Type definitions and strict type definitions d. Program control structures and data storage i. Looping structures (For Loops and While Loops) a. Indexing on loop boundaries b. Shift registers ii. Case and Sequence structures a. Flat and Stacked sequence structures b. Case selector values and data types c. Data passing—tunnels and sequence locals iii. Event structures a. Notify and filter events (user interface) b. Value (Signaling) properties of controls c. Dynamic events and user events iv. Formula Node v. Conditional Disable and Diagram Disable structures vi. Timed structures vii. Local, global, and shared variables

4. Programming VIs and functions	a. Numeric, Boolean, string, path, and variant b. Conversion, comparison, and manipulation c. Arrays and clusters d. Timing i. Wait timers, Tick Count (ms), and Date/Time functions ii. Timing functions related to Timed structures e. ASCII, binary, datalog, storage (`.tdm`), waveform, XML, and configuration file I/O formats f. Waveform and waveform file I/O g. Dynamic and User events
5. Data communication, synchronization	a. Local, global, and shared variables b. DataSocket c. TCP and UDP d. Synchronization i. Notifiers ii. Queues iii. Semaphores
6. VI Server	a. Configuring the VI Server b. Class hierarchy, references, Property Nodes, and Invoke Nodes c. Dynamically loading VIs
7. Error handling VIs and functions	a. Error clusters b. Dialog & User Interface VIs c. Custom error codes
8. Design patterns	a. Simple state machine b. User interface event handler c. Queued message handler d. Producer/consumer (data) and producer/consumer (events) e. Functional global variables
9. SubVI design	a. SubVI creation methods b. Connector panes and connection types c. Polymorphic subVIs d. Options related to subVIs e. Error handling
10. Debugging tools and techniques	a. Debugging tools i. **Error list** window ii. Execution highlighting iii. Breakpoints and single stepping iv. Generic and custom probes b. Debugging practices and techniques for different situations

11. VI design and documentation	a. Refer to the *LabVIEW Style Checklist* topic of the *LabVIEW Help* for information on the following items i. User interface design and block diagram layout ii. Modular and hierarchical design iii. SubVI icons and connector pane layout (standard) iv. VI properties v. Documenting VIs
12. Memory, performance and determinism	a. Tools for identifying memory and performance issues i. Profile memory and performance ii. Show buffer allocations iii. VI metrics b. Programming practices i. Enforcing dataflow ii. User interface updates and response to user interface controls iii. Data type selection, coercion, and buffer allocation iv. Array, string, and loop operations v. Local and global variables, Property Nodes, and references

CLAD Topics Details

1. **LabVIEW programming principles:**
   a. Data flow
      1. Define data flow
      2. Identify the importance of data flow in LabVIEW
      3. Identify programming practices that enforce data flow in the block diagram, VIs, and subVIs
      4. Identify programming practices that break data flow
   b. Polymorphism
      1. Define polymorphism
      2. Identify the benefits of polymorphism
      3. Determine the output or intermediate values of data elements in a VI that utilizes polymorphic inputs

2. **LabVIEW Environment:**
   a. Front panel window, block diagram, and connector pane
      1. Identify the relationship between front panel window and block diagram objects in a VI and their connections through the connector pane
      2. Identify which types of VIs do not have a block diagram
      3. Identify the purpose of the connector pane and icon
   b. Palettes
      1. Identify the type of palettes and their function
   c. Configuration options
      1. Identify the impact of configuration options for the following items
         a) Front panel window
         b) Block diagram
         c) Environment

3. **Front panel window and block diagram objects, data types, variables, and software constructs:**
   a. Front panel window and block diagram objects
      1. Select the most appropriate object for the front panel window of an application
      2. Describe the connection between an object on the front panel window and its terminals
      3. Select between an object on the front panel window or a block diagram constant
      4. Select the most appropriate palette(s), legends(s) and update mode for graphs and charts
      5. Determine the most appropriate mechanical action for Boolean controls
      6. Determine the appropriateness of Property Nodes and select the appropriate property
   b. Data Types and data structures
      1. Select the most appropriate data type for front panel window and block diagram objects

2. Select the most appropriate method to group related data items
3. Describe the waveform data type and use it to display data on graphs and charts
4. Describe the timestamp data type and use it to time stamp measurement data
5. Identify the applications in which the variant data type is the most appropriate choice.

c. Working with front panel window objects and data types
   1. Determine the most appropriate representation, range, format, precision, and scaling to represent a data item
   2. Identify and describe the scenarios in which you would need to customize a control
   3. Distinguish between a type definition and a strict type definition
   4. Identify and describe the applications which would benefit from the use of a type definition or a strict type definitions
   5. Determine if a type definition or a strict type definition is needed to represent a data item

d. Program control structures and data storage
   1. Select and apply the most suitable program control structure
   2. Select and implement a data storage mechanism for a program control structure
   3. Identify and describe the function of looping structure components
   4. Select a While Loop or For Loop as the most appropriate looping structure
   5. Describe auto-indexing and determine the effects of turning indexing on or off with each type of looping structure
   6. Determine the data values in a loop that utilizes auto-indexing, after a set number of iterations occur or upon loop termination
   7. Describe the use and initialization of shift registers as data storage elements
   8. Determine the data values in the shift register(s) after a set number of iterations occur or upon loop termination
   9. Identify the pros and cons and select between a Sequence structure or Case structure
   10. Identify the pros and cons between Flat and Stacked Sequence structures with respect to data flow and data passing
   11. Select the most appropriate data type to wire to the selector terminal of a Case structure
   12. Identify two types of output tunnels in a Case structure and identify the pros and cons of each type
   13. Identify the advantages of Event structures for event-driven programming
   14. Identify the components of an Event structure
   15. Identify the different ways in which an event may be generated
   16. Identify the different events that an Event structure can handle
   17. Identify the two types of user interface events
   18. Recognize the impact of locking the front panel window for user interface events
   19. Identify and explain the application need for dynamic events
   20. Apply the techniques to register and un-register dynamic events

21. Identify and explain the application need for user events
22. Apply the techniques to register, generate, and destroy user events
23. Determine the most appropriate event mechanism for an application
24. Identify the components of a Formula Node and the relationship between the script variables and input and output terminals
25. Determine the output of an application that utilizes a Formula Node
26. Identify the difference(s) between Conditional Disable and Diagram Disable structures
27. Determine the output of an application that utilizes a Conditional Disable or Diagram Disable structure
28. Determine if a Conditional Disable or Diagram Disable structure is most appropriate for an application
29. Select the most appropriate Timed structure
30. Identify and configure the input and outputs of the components of different Timed structures
31. Configure the priority of Timed structures
32. Determine the output of an application that consists of multiple Timed structures with set priorities
33. Identify the difference(s) between shared, local, and global variables

4. Programming VIs, functions, and properties

a. VIs and Express VIs

1. Determine the output or intermediate values of data elements in an application that utilizes VIs and functions from the following list
2. Utilizing VIs and functions from the following list, determine the most appropriate VI(s) or function(s) to complete a specified functionality

List of the VIs and functions that apply this section:

a) Numeric—**Numeric**, **Conversion**, **Data Manipulation**, and **Comparison** palettes
b) Boolean—**Boolean** palette
c) String—**String**, **String / Number Conversion**, and **String /Array /Path Conversion** palettes
d) Path—Path functions on the **File I/O** palette
e) Variants—Variant functions on the **Cluster & Variant** palette
f) Array—**Array** palette
g) Cluster—Cluster functions on the **Cluster & Variant** palette
h) Timing—**Timing** and **Timed Structures** palettes
i) File I/O—**File I/O** and **XML** palettes
j) Waveform—**Waveform** palette
k) Events—**Events** palette

5. Data Communication and synchronization VIs and functions

a. Functions, VIs, and Express VIs

1. Select the most appropriate method to communicate data between applications
2. Identify the pros and cons of and use local, global, or shared variables for data communication

3. Identify and explain the different methods of communication using TCP and UDP
4. Identify and explain the difference(s) between notifiers and queues
5. Select the most appropriate method to communicate data between multiple block diagram sections
6. Identify the applications that use semaphores for data protection and synchronization
7. Determine the output or intermediate values of data elements in an application that utilizes VIs and functions from the following list
8. Utilizing VIs and functions from the following list, determine the most appropriate VI(s) or function(s) that are needed to complete a specified functionality

List of the VIs and functions that apply this section:

a) DataSocket—**DataSocket** palette
b) TCP and UDP—**TCP** and **UDP** palettes
c) Notifiers—**Notifier Operations** palette
d) Queues—**Queue Operations** palette
e) Semaphores—**Semaphore** palette

6. VI Server

a. Configuration
   1. Apply appropriate settings for configuring the VI Server

b. Class hierarchy, references, Property Nodes, Invoke Nodes, and dynamically loading VIs
   1. Identify the different methods of dynamically loading and running VIs, the reference type, properties, and methods used to support the method
   2. Given a class hierarchy, recognize property and method inheritance and use reference type casting VIs to obtain a reference to a higher or lower class in the hierarchy
   3. Determine the most appropriate method for dynamically loading and running VIs

7. Error handling VIs and Functions

a. Error clusters and wires
   1. Identify the components of error clusters and terminals that accept the error wire.
   2. Identify the differences between errors and warnings
   3. Design VIs that adhere to the *LabVIEW Style Checklist* topic of the *LabVIEW Help*. For example, utilize error checking to control While Loops, handle errors with Case structures, and use appropriate terminals on the connector pane

b. Error and Dialog VIs
   1. Given a VI or subVI, identify the most appropriate locations to handle errors and notify the user or a calling VI of the error

2. Utilizing VIs and functions from the **Dialog & User Interface** palette, determine the most appropriate VI(s) or function(s) to complete a specified error handling and reporting functionality

c. Custom error codes
   1. Identify the numeric range and methods to define custom error codes and use the custom error codes to generate errors from VIs

8. Design patterns

a. Select a design pattern:
   1. Identify a design pattern, explain its pros and cons, and compare it with other design patterns
   2. Given an application requirement, select the most appropriate design pattern from the following:
      a) Simple state machine
      b) User interface event handler
      c) Queued message handler
      d) Producer/consumer (data)
      e) Producer/consumer (events)
      f) Functional global variable

9. SubVI Design

a. Methods to create subVIs
   1. Identify and explain the methods used to create subVIs and the pros and cons of each method

b. Connector pane and connection types
   1. Select the most appropriate connector pane and assign terminals according to recommendations in the *LabVIEW Style Checklist* topic of the *LabVIEW Help*
   2. Identify which terminals are Required, Recommended, or Optional
   3. Given a requirement, identify which terminals to set as Required, Recommended, or Optional connections

c. Polymorphic SubVIs
   1. Evaluate if a polymorphic subVI design is the most appropriate choice
   2. Identify the pros, cons, and restrictions in developing a polymorphic subVI

d. Options related to SubVIs
   1. Identify the execution and window settings and explain the implication of each setting
   2. Select and apply the most appropriate execution and window settings for all or a single instance of a subVI

e. Error handling
   1. Apply error handling to a subVI as recommended in the *LabVIEW Style Checklist* topic of the *LabVIEW Help*

10. Debugging tools and techniques

a. Debugging tools
   1. Identify and explain the implications of the VI Properties settings that determine how LabVIEW handles errors and warnings

2. Identify the errors in a VI that result in a broken **Run** button and use the **Error list** window to determine the cause
3. Explain the use of execution highlighting for tracing data flow and as a helping tool in conjunction with other debugging tools
4. Explain and apply breakpoints, execution suspension, and single stepping tools to debug VIs and subVIs
5. Utilize the Probe tool, indicators, generic probes, conditional probes, and custom probes to display data values

b. Debugging practices and techniques for different situations
   1. Given an error situation, select the most appropriate method to debug the error
   2. Determine if a given block diagram would result in an error condition

11. VI design and documentation

a. Utilize and apply the *LabVIEW Style Checklist* topic of the *LabVIEW Help* for the following:
   1. User interface design and block diagram layout
   2. Modular and hierarchical design
   3. SubVI icons and connector pane layout (standard)
   4. VI Properties
   5. Documenting VIs

12. Memory, performance and determinism

a. Tools for identifying memory and performance issues
   1. Select the most appropriate tool(s) for identifying memory and performance issues

b. Programming practices
   1. Identify block diagram code that breaks data flow and techniques that enforce data flow
   2. Identify block diagram code that can slow down user interface response or update and identify techniques to improve the response
   3. Select the most appropriate data type to limit coercion, buffer allocations, and optimize speed and memory reuse
   4. Identify array, string, and loop operations that limit memory and performance and identify techniques and methods that optimize performance
   5. Identify race conditions, memory and performance issues associated with the use of local and global variables, Property Nodes, and references, and use techniques to optimize their use

CLAD Exam Preparation Resources

Use the following resources for exam preparation:

CLAD Preparation Course:

- National Instruments CLAD Preparation Course (Online) webcast

CLAD Sample Exam:

- CLAD Sample Exam

Training / Tutorials:

- Online LabVIEW Basics
- LabVIEW Introduction Course – Three Hours
- LabVIEW Introduction Course – Six Hours
- National Instruments LabVIEW Basics I and LabVIEW Basics II courses:
  - Instructor-led
  - Self- paced using the course manuals
- National Instruments LabVIEW Intermediate I and Intermediate II courses:
  - Instructor-led
  - Self- paced using the course manuals
- National Instruments LabVIEW Upgrade Primer course (online)

Web Resources:

- LabVIEW Development Guidelines
- Free practice LabVIEW Fundamentals Exam
- National Instruments Academic Web
- National Instruments Developer Zone
- National Instruments Developer (LabVIEW) Zone
- National Instruments LabVIEW Zone
- National Instruments LabVIEW Support
- LabVIEW Manuals Online (current manuals)
- Free tutorials:
  - LabVIEW Review (Colorado School of Mines)
  - LabVIEW Tutorial (University of Sydney)
  - LabVIEW for Dummiees© (Illinois Institute of Technology)
  - LabVIEW Tutorial (University of Buffalo)
  - LabVIEW Tutorial Series (University of Western Australia)