

1. Navigating Labview.

LabVIEW. Labview is a graphical programming environment you can use to develop sophisticated measurement, test, and control systems.

PROJECT EXPLORER. Projects in LabVIEW consist of VIs, files necessary for those VIs to run properly, and supplemental files such as documentation or related links. Use the Project Explorer window to manage projects in LabVIEW. The Project Explorer window includes the following items by default:

- **Project root:** Contains all other items in the Project Explorer window. The label on the project root includes the filename for the project.
- **My Computer:** Represents the local computer as a target in the project.
- **Dependencies:** Includes VIs and items that VIs under a target require.
- **Build Specifications:** Includes build configurations for source distributions and other types of builds available in LabVIEW toolkits and modules.

FOLDERS IN PROJECT EXPLORER. There are two types of folders in the project: Auto-populated folders and Virtual folders.

- **Auto-populated folders:** adds a directory on disk to the project.
- **Virtual Folder:** organizes project items and doesn't represent files on disk.

LABVIEW FILES. The common labview extensions files are Labview Project (.lvproj), Virtual Instrument (.vi) and Custom Control (.ctl).

PARTS OF A VI. Labview VIs contain three main components: front panel, block diagram, and the icon/connector pane.

- **Front panel:** User interface for the VI. Contains controls and indicators, which are the interactive input and output terminals of the VI, respectively. LabVIEW has different control palettes with objects for building user interfaces. These palettes include the Modern, Silver, Classic, and Systems palettes.
- **Block diagram:** Block diagram objects include terminals, subVIs, functions, constants, structures, and wires, which transfer data among other block diagram objects. The components of a block diagram are: terminals, nodes, functions, nodes, SubVIs, icon, connector pane.
- **Icon/connector pane:** Icons are the graphical representation of VIs and the connector pane is the map of inputs and outputs of a VI.

EXPRESS VIs. Express VIs are a type of subVI that you configure with dialog boxes. Icons for Express VIs appear on the block diagram as icons surrounded by a blue field.

WIRES. Wires transfer data between block diagram objects. A broken wire appears as a dashed black line with a red X in the middle. You cannot run a VI that contains a broken wire. Common wire types: scalar, 1D Array and 2D Array.

CONTEXT HELP. The Context Help window displays basic information about LabVIEW objects when you move the cursor over each object.

CONTROLS PALETTE. The Controls palette contains the controls and indicators you use to create the front panel.

FUNCTION PALETTE. The Functions palette contains the VIs, functions and constants you use to create the block diagram.

QUICK DROP. The Quick Drop dialog box lets you quickly find controls, functions, VIs, and other items by name.

2. Creating your first application.

DATAFLOW. Dataflow is the movement of data through the nodes of a block diagram.

LABVIEW DATA TYPES. Block diagram terminals visually communicate information about the data type represented. Terminal colors, text, arrow direction, and border thickness all provide visual information about the terminal.

- **Boolean Data:** Boolean controls and indicators to enter and display Boolean (TRUE/FALSE) values. LabVIEW stores Boolean data as 8-bit values. If the 8-bit value is zero, the Boolean value is FALSE. Any nonzero value represents TRUE.
- **NUMERIC DATA:** LabVIEW represents numeric data types as floating-point numbers, fixed-point numbers, integers, unsigned integers, and complex numbers. The difference among the numeric data types is the number of bits they use to store data and the data values they represent.
- **STRINGS:** A string is a sequence of displayable or non-displayable ASCII characters. Strings provide a platform-independent format for information and data.
- **ENUMS:** Enums give users a list of items from which to select. Enums are useful because they make strings equivalent to numbers, which are easy to manipulate on the block diagram. Each item represents a pair of values: a string value and a 16-bit integer.
- **DYNAMIC:** Stores the information generated or acquired by an Express VI.
- **PATH:** Stores the location of a file or directory using the standard syntax for the platform you are using.
- **WAVEFORM:** Carries the data, start time, and dt of a waveform.



Activity 2-1: Exploring Dataflow

Goal: Understand how dataflow determines the execution order in a VI.

Description: Work through this exercise on your own first, and then as a class, we will discuss how data flow determines execution order. You can find answers to these questions following this section.



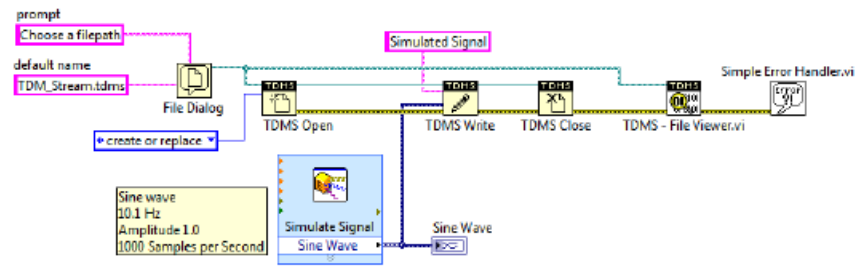
Note Nodes are objects on the block diagram that have inputs and/or outputs and perform operations when a VI runs.

Using Figure 2-1, answer questions 1 through 5.

1. Which node executes first? Is there any dependency between the File Dialog function and the Simulate Signal Express VI?
2. Which node executes last?
3. Because a wire connects the File Dialog function to the TDMS File Viewer VI, can the TDMS File Viewer VI execute before the TDMS Close function?
4. How many nodes must execute before the TDMS Write function can execute?

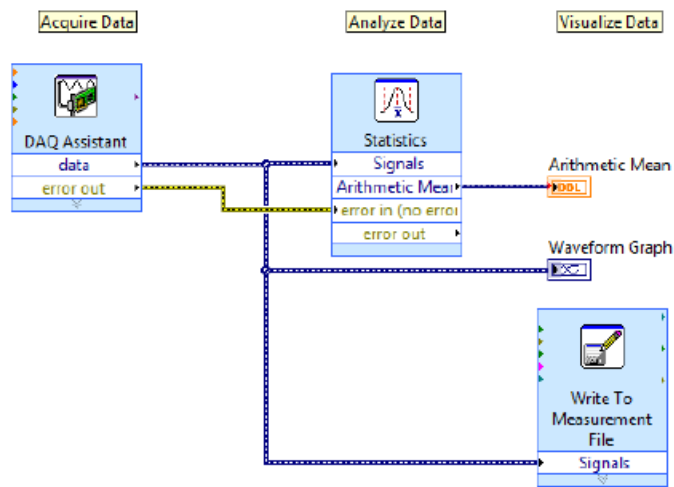
5. Should a well-designed block diagram flow in a particular direction?

Figure 2-1. Dataflow: Example A



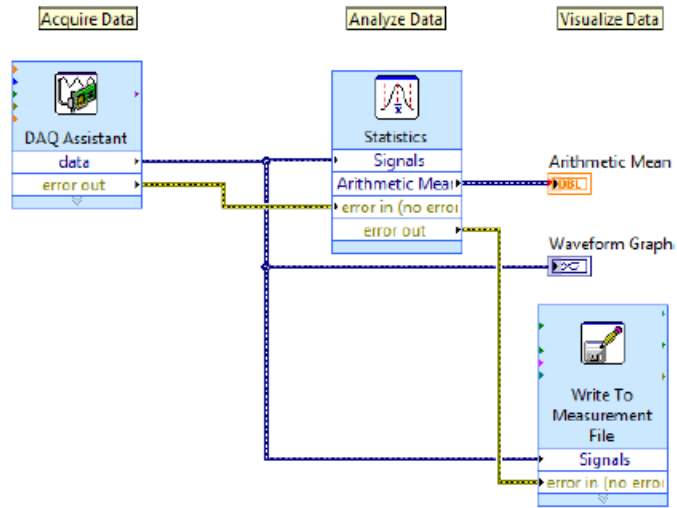
6. In Figure 2-2, which Express VI executes last?

Figure 2-2. Dataflow: Example B



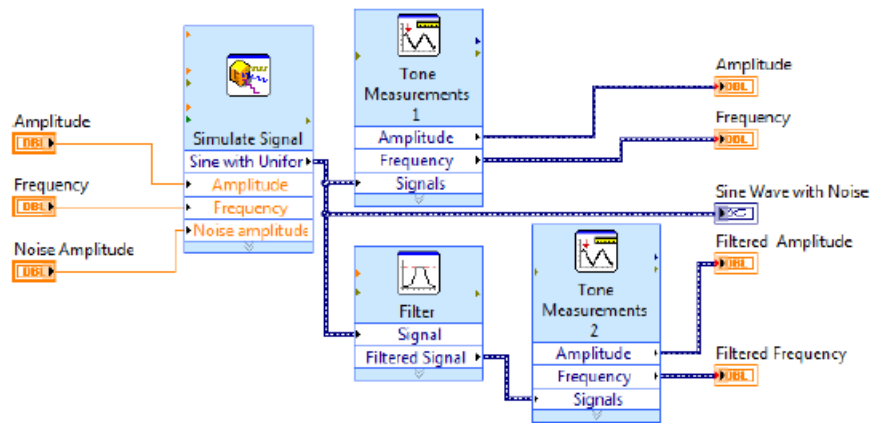
7. In Figure 2-3 an error wire connects the Express VIs. Which Express VI executes last?

Figure 2-3. Dataflow: Example C



8. In Figure 2-4, which Tone Measurements Express VI executes last?

Figure 2-4. Dataflow: Example D





Activity 2-1: Exploring Dataflow - Answers

1. Which node executes first? Is there any dependency between the File Dialog function and the Simulate Signal Express VI?

Either the File Dialog function or the Simulate Signal Express VI can execute first. There is no data dependency between the two nodes so either of them can execute first or they can execute simultaneously.

2. Which node executes last?

The last node to execute is the Simple Error Handler VI.



Note Terminals are not considered nodes.

3. Because a wire connects the File Dialog function to the TDMS File Viewer VI, can the TDMS File Viewer VI execute before the TDMS Close function?

No. The TDMS File Viewer VI cannot execute before the TDMS Close function because the yellow error wire connecting the TDMS Close function and the TDMS File Viewer VI forces data dependency. Remember, the data to all inputs of a node must be available before a node can execute. Therefore, the TDMS File Viewer VI must receive data from both the green Boolean wire and the yellow error wire before the VI can execute.

4. How many nodes must execute before the TDMS Write function can execute?

Three nodes must execute before the TDMS Write function can execute: File Dialog, TDMS Open, and Simulate Signal. The TDMS Write function also depends on the Simulated Signal string constant, but that input is instantaneous.

5. Should a well-designed block diagram flow in a particular direction?

Yes. A well-designed block diagram typically flows from left to right. This makes it easier to see the flow of data on the block diagram. However, do not assume left-to-right or top-to-bottom execution when no data dependency exists.

6. In Figure 2-2, which Express VI executes last?

Either the Statistics Express VI or the Write to Measurement File Express VI executes last or they execute in parallel. The DAQ Assistant Express VI cannot execute last because both the Statistics Express VI and the Write to Measurement File Express VI are dependent on the data signal from the output of the DAQ Assistant Express VI.



Note In LabVIEW, the flow of data, rather than the sequential order of commands, determines the execution order of block diagram elements. Therefore, it is possible to have simultaneous operations.

7. In Figure 2-3 an error wire connects the Express VIs. Which Express VI executes last?

The Write to Measurement File Express VI executes last. It has a data dependency on both the DAQ Assistant Express VI and the Statistics Express VI.

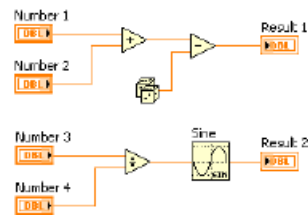
8. In Figure 2-4, which Tone Measurements Express VI executes last?

Either one of the Tone Measurement Express VIs can execute last. Even though the Tone Measurements 2 Express VI has an extra dependency on the Filter Express VI, the Filter Express VI might execute before the Tone Measurements 1 Express VI allowing the Tone Measurements 2 Express VI to execute before the Tone Measurements 1 Express VI. Although it seems as if the Tone Measurements 1 Express VI would execute first, without an explicit data dependency there is no way to know definitely it would execute first.



Activity 2-3: Lesson Review - Answers

Refer to the figure below to answer the following quiz questions.



1. Which function executes first: Add or Subtract?
 - a. Add
 - b. Subtract
 - c. Unknown
2. Which function executes first: Sine or Divide?
 - a. Sine
 - b. Divide
 - c. Unknown
3. Which function executes first: Random Number, Divide or Add?
 - a. Random Number
 - b. Divide
 - c. Add
 - d. Unknown
4. Which function executes last: Random Number, Subtract or Add?
 - a. Random Number
 - b. Subtract
 - c. Add
 - d. Unknown
5. If an input to a function is marked with a red dot (known as coercion dot), what does the dot indicate?
 - a. Data was transferred into a structure.
 - b. That input has not been wired
 - c. The wire is broken
 - d. The value passed into a node was converted to a different representation.
6. Which mechanical action causes a Boolean control in the FALSE state to change to TRUE when you click it and stay TRUE until LabVIEW has read the value?
 - a. Switch Until Released
 - b. Switch When Released
 - c. Latch When Pressed
 - d. Latch When Released

3. TROUBLESHOOTING AND DEBUGGING VIs

Breakpoints. Use the Breakpoint tool to place a breakpoint on a VI, node, or wire and pause execution at that location.

Retain wire values. Retain Wire Values enables you to probe wire values after execution completes. This option uses more memory because LabVIEW is storing data for each wire as it runs.

Undefined or unexpected data. Floating-point operations return the following two symbolic values that indicate faulty computations or meaningless results.

Error handling. Anticipation, detection, and resolution of warnings and errors.

Automatic error handling. At run time, LabVIEW suspends execution, highlights node where error occurred, and displays Error dialog box.

Manual error handling. You control when dialog boxes appear, propagate errors through error in/error out clusters, terminate error chain with Simple Error Handler.

Error clusters. VIs and functions return errors in one of two ways—with numeric error codes or with an error cluster. Typically, functions use numeric error codes, and VIs use an error cluster, usually with error inputs and outputs. Use the error cluster controls and indicators to create error inputs and outputs in subVIs. The components of an error cluster are:

1. status: a Boolean value that reports TRUE if an error occurs
2. code: a 32-bit signed integer that identifies the error numerically. A non-zero error code is coupled with a status of FALSE signals a warning rather than an error
3. source: a string that identifies where the error occurred

Differences between errors and warnings.

- Errors: Status = True, Code = Non-zero, more severe and passed to the next node without executing that part of the code.
- Warnings: Status = False, Code = Non-zero, less severe and node executes normally, but it is important to monitor warnings during development to ensure proper behavior of your application.

Merge Errors. The Merge Errors Function Returns the first error found. If no error is found, it returns the first warning and does not concatenate errors. Use Merge Errors to propagate errors along wires and merge errors from different wire paths.



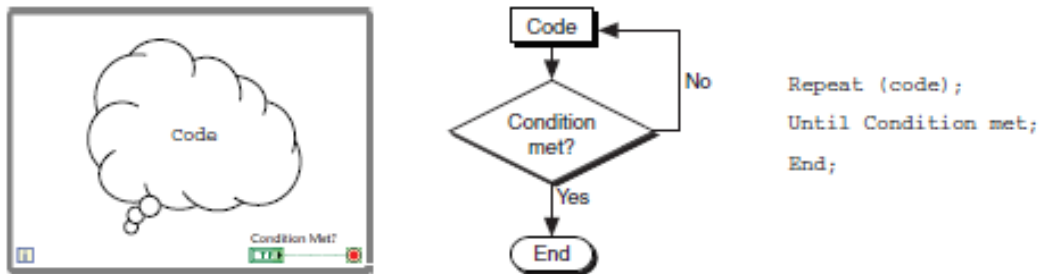
Activity 3-2: Lesson Review - Answers

1. Which of the following will result in a broken run arrow?
 - a. **A subVI is broken**
 - b. The diagram includes a divide by zero
 - c. A required subVI input is unwired
 - d. A Boolean terminal is wired to a numeric indicator
2. Which of the following are components and data types of the error cluster?
 - a. **Status: Boolean**
 - b. Error: String
 - c. **Code: 32-bit integer**
 - d. **Source: String**
3. All errors have negative error codes and all warnings have positive error codes.
 - a. True
 - b. **False**
4. Merge Errors function concatenates error information from multiple sources.
 - a. True
 - b. **False**

4. Using Loops.

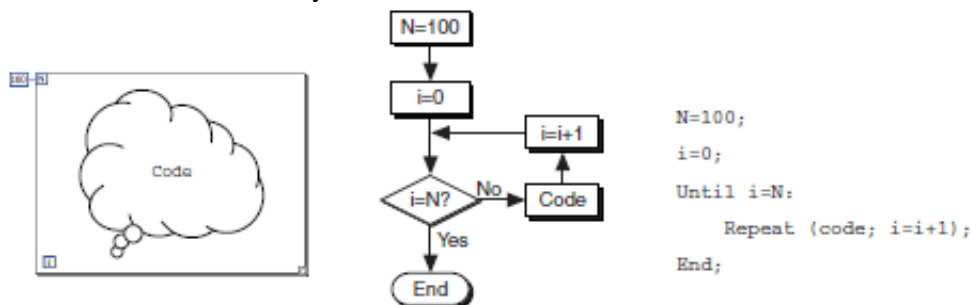
While Loops. Similar to Do Loop or a Repeat-Until Loop

- Repeats code segment until a condition is met
- Always execute at least once
- Iteration terminal—output that contains the number of complete iterations
- Iteration terminal always starts at zero
- Conditional terminal sets the condition for stopping the loop



For loops. Repeats code a certain number of times

- The value in the count terminal (an input terminal) indicates how many times to repeat the subdiagram in the For Loop.
- Can execute zero times
- Iteration terminal—output that contains the number of complete iterations
- Iteration terminal always starts at zero





Activity 4-1: While Loops vs. For Loops - Answers

Scenario 1

Acquire pressure data in a loop that executes once per second for one minute.

1. If you use a While Loop, what is the condition that you need to stop the loop?

While Loop: Time = 1 minute

2. If you use a For Loop, how many iterations does the loop need to run?

For Loop: 60 iterations

3. Is it easier to implement a For Loop or a While Loop?

Both are possible

Scenario 2

Acquire pressure data until the pressure is greater than or equal to 1400 psi.

1. If you use a While Loop, what is the condition that you need to stop the loop?

While Loop: Pressure = 1400 psi

2. If you use a For Loop, how many iterations does the loop need to run?

For Loop: unknown

3. Is it easier to implement a For Loop or a While Loop?

A While Loop. Although you can add a conditional terminal to a For Loop, you still need to wire a value to the count terminal. Without more information, you do not know the appropriate value to wire to the count terminal.

Scenario 3

Acquire pressure and temperature data until both values are stable for two minutes.

1. If you use a While Loop, what is the condition that you need to stop the loop?

While Loop: [(Last Temperature = Previous Temperature) for 2 minutes or more] and [(Last Pressure = Previous Pressure) for 2 minutes or more]

2. If you use a For Loop, how many iterations does the loop need to run?

For Loop: unknown

3. Is it easier to implement a For Loop or a While Loop?

A While Loop. Although you can add a conditional terminal to a For Loop, you still need to wire a value to the count terminal. Without more information, you do not know the appropriate value to wire to the count terminal.

Scenario 4

Output a voltage ramp starting at zero, increasing incrementally by 0.5 V every second, until the output voltage is equal to 5 V.

1. If you use a While Loop, what is the condition that you need to stop the loop?

While Loop: Voltage = 5 V

2. If you use a For Loop, how many iterations does the loop need to run?

For Loop: 11 iterations (Including the two end points, count the iteration for each value - 0, 0.5, 1.0, 1.5, ... 4.5, 5.0.)




3. Is it easier to implement a For Loop or a While Loop?

Both are possible.

Tunnels. Tunnels transfer data into and out of structures. Data pass out of a loop after the loop terminates.

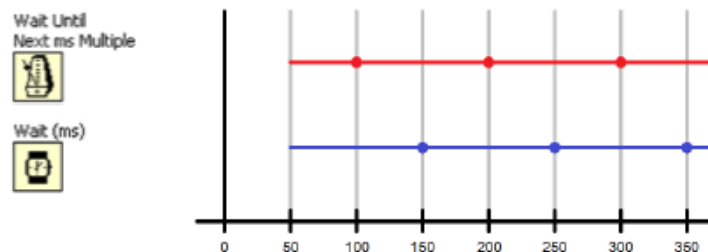
Condition terminal. a conditional terminal to configure a For Loop to stop when a Boolean condition or an error occurs. A For Loop with a conditional terminal executes until the condition occurs or until all iterations are complete, whichever happens first.

Timing a VI. Use a wait function inside a loop to accomplish the following actions:

Wait Function	Behavior
Wait Until Next ms Multiple 	Monitors a millisecond counter and waits until the millisecond counter reaches a multiple of the amount you specify. This function is synced to the system clock.
Wait (ms) 	Waits until the millisecond counter counts to an amount equal to the input you specify
Time Delay Express VI 	Similar to the Wait (ms) function with the addition of built-in error clusters.

Wait Function Timing

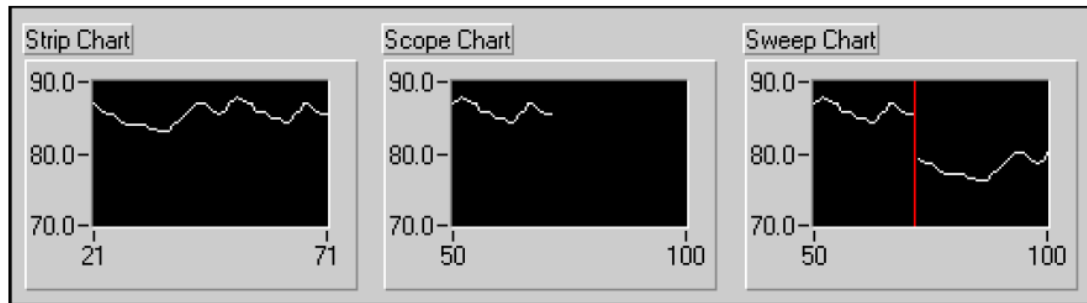
The following illustration compares the differences between two common timing functions: Wait Until Next ms Multiple and Wait (ms). This timing diagram assumes that the wait functions begin running immediately for each loop iteration and that the loop is ready to iterate as soon as the wait function finishes.



Shift registers. Shift registers store data values from previous iterations of a loop in LabVIEW.

Waveform data. The waveform chart is a special type of numeric indicator that displays one or more plots of data typically acquired at a constant rate. Waveform charts can display single or multiple plots. You can configure how the chart updates to display new data. The chart uses the following modes to display data:

- **Strip Chart**—Shows running data continuously scrolling from left to right across the chart with old data on the left and new data on the right.
- **Scope Chart**—Shows one item of data, such as a pulse or wave, scrolling partway across the chart from left to right.
- **Sweep Chart**—Works similarly to a scope chart except it shows the old data on the right and the new data on the left separated by a vertical line.



Activity 4-2: Lesson Review - Answers

1. Which structure must run at least one time?
 - a. **While Loop**
 - b. For Loop

5. Creating and leveraging data structures.

Arrays. Collection of data elements that are of the same type.

Elements. The data that make up the array. Elements can be numeric, Boolean, path, string, waveform, and cluster data types.

Dimension. Length, height, or depth of the array. Arrays can have one or more dimensions and as many as (231)-1 dimensions.

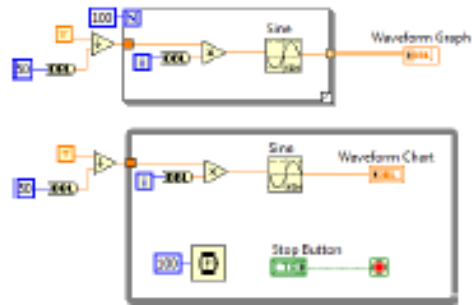
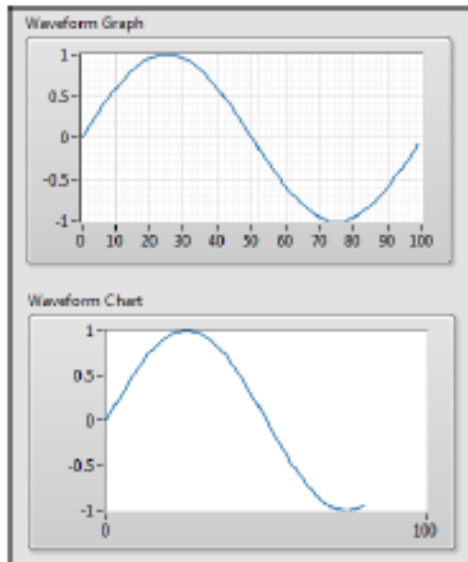
Arrays restrictions. You cannot create arrays of arrays. However, you can use a multidimensional array or create an array of clusters where each cluster contains one or more arrays. Also, you cannot create an array of subpanel controls, tab controls, .NET controls, ActiveX controls, charts, or multi-plot XY graphs.

Polymorphism. The ability of VIs and functions to automatically adapt to accept input data of different data types. Functions are polymorphic to varying degrees—none, some, or all of their inputs can be polymorphic.

Auto indexing. The ability to automatically process every element in an array.

Waveform graphs. A waveform graph collects the data in an array and then plots the data to the graph.

Waveform charts. Charts are generally used inside the While Loop and graphs are generally outside the While Loop.



Auto indexing with a conditional tunnel. You can determine what values LabVIEW writes to the loop output tunnel based on a condition you specify.

Auto indexing input. Use an auto-indexing input array to perform calculations on each element in an array. If you wire an array to an auto-indexing tunnel on a For Loop, you do not need to wire the count (N) terminal.

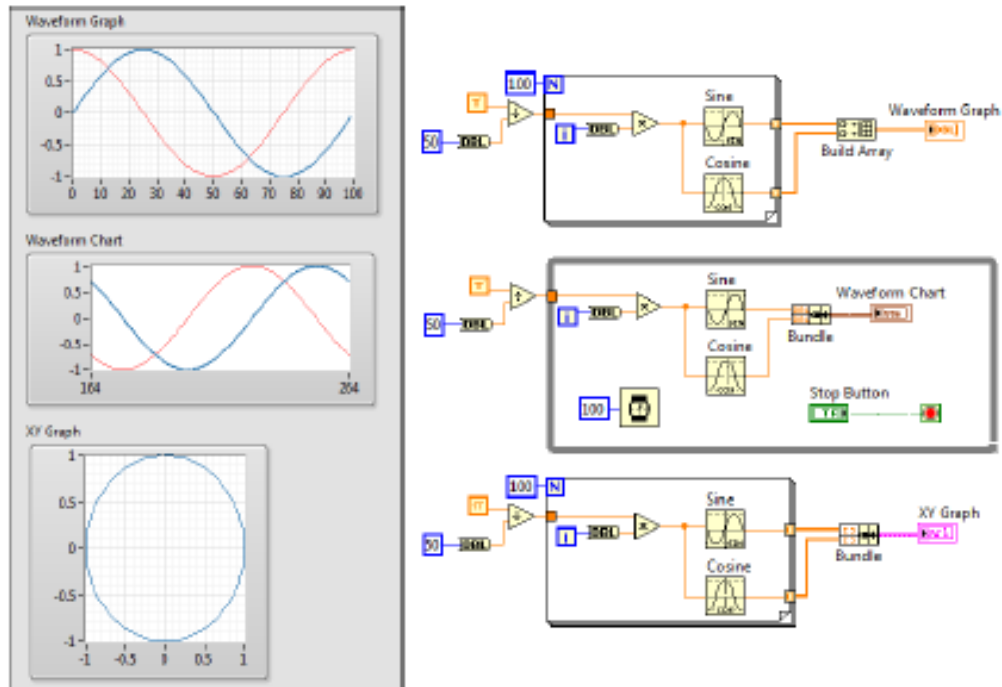
Auto indexing input-different array sizes. If the iteration count terminal is wired and arrays of different sizes are wired to auto-indexed tunnels, the actual number of iterations becomes the smallest of the choices.

Clusters. Clusters group data elements of mixed types controls or indicators it can't be mixed of control and indicators.

Differences between clusters and arrays.

- Clusters: mixed data types and fixed size.
- Arrays: contain only one data type and vary in size.

XY graph. The Bundle function is often used to create multi-chart plot charts and XY plots. The Build Array function is used to create multi-plot waveform graphs.



Type definitions. Three types of custom controls—Control, Type Definition, Strict Type Definition.

- Custom controls are templates and used as starting points for other similar controls. Changes made to one control does not reflect in other controls.
- Type definitions and strict type definitions link to all the instances of a custom control or indicator to a saved custom control or indicator file. You can make changes to all instances of the custom control or indicator by editing only the saved custom control or indicator file
- Strict type defs apply cosmetic changes too, whereas type defs do not.



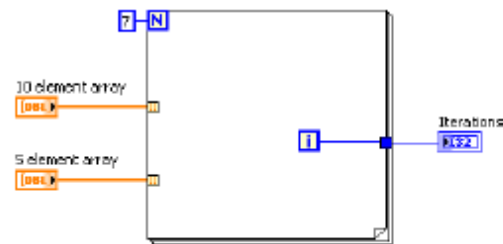
Activity 5-2: Lesson Review - Answers

1. You can create an array of arrays.

- a. True
- b. False

You cannot drag an array data type into an array shell. However, you can create two-dimensional arrays.

2. You have two input arrays wired to a For Loop. Auto-indexing is enabled on both tunnels. One array has 10 elements, the second array has five elements. A value of 7 is wired to the Count terminal, as shown in the following figure. What is the value of the Iterations indicator after running this VI?



Value of Iterations = 4

LabVIEW does not exceed the array size. This helps to protect against programming error. LabVIEW mathematical functions work the same way—if you wire a 10 element array to the x input of the Add function, and a 5 element array to the y input of the Add function, the output is a 5 element array.

Although the for loop runs 5 times, the iterations are zero based, therefore the value of the Iterations indicators is 4.

3. Which of the following custom control settings defines the data type of all instances of a control but allows for different colors and font styles?

- a. Control
- b. Type Definition
- c. Strict Type Definition
- d. Cluster control

4. You have input data representing a circle: X Position (I16), Y Position (I16), and Radius (U32). In the future, you might need to modify your data to include the color of the circle (U32).

What data structure should you use to represent the circle in your application?

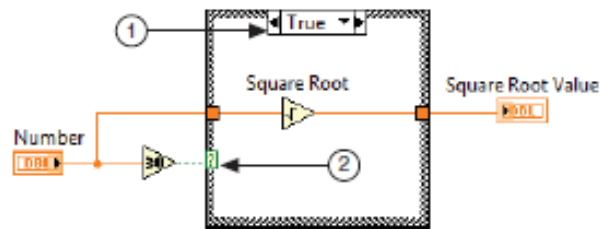
- a. Three separate controls for the two positions and the radius.
- b. A cluster containing all of the data.
- c. A custom control containing a cluster.
- d. A type definition containing a cluster.
- e. An array with three elements.

6. Case structures.



Activity 6-1: Case Structures Review - Answers

Figure 6-2. Case Structures Review: Answers



- | 1 Case Selector Label | 2 Selector Terminal |
|--|---------------------|
| 1. What is the purpose of the Case Structure? | |
| a. Execute one of its subdiagrams based on an input value | |
| b. Repeat a section of code until a condition occurs | |
| c. Execute a subdiagram a set number of times | |
| 2. How many of its cases does a Case Structure execute at a time? | |
| a. All of them | |
| b. One | |
| 3. What is the purpose of the Case Selector Label? | |
| a. Lets you wire an input value to determine which case executes | |
| b. Show the name of the current state and enable you to navigate through different cases | |
| 4. What is the purpose of the Selector Terminal? | |
| a. Lets you wire an input value to determine which case executes | |
| b. Show the name of the current state and enable you to navigate through different cases | |

Selector terminal data types.

Data Type	Example
Boolean A newly-created Case structure defaults to a Boolean input. The Case Structure includes a True case and a False Case.	
Integer Case Structure has any number of cases. Specify a Default case. The numeric representation of the integer input will determine the range of possible values for the Case Selector Label. You can specify ranges of values for the Case Selector Label. Use Radix option in shortcut menu to specify whether the Case Selector Label displays values in decimal, hexadecimal, octal, or binary.	
String Case Structure has any number of cases. Specify a Default case. By default, string values are case sensitive. Shortcut menu includes option for Case Insensitive Match for the string text.	

Data Type	Example
Enum Possible to ensure that the Case Structure includes a case for every item in the enum. Right-click the border of the Case Structure and select Add Case for Every Value to create a case for every item.	
Error Cluster Case Structure includes an Error Case and a No Error Case. Wire an error cluster to the terminal to execute code if there is no error and skip code if there is an error.	

Event driven programming. Method of programming where the program waits on an event to occur before executing the code written to handle that event.

Event. An asynchronous notification that something has occurred. Events can originate from the user interface, external I/O, or other parts of the program. In this course, you will only learn about user interface events, which include mouse clicks, key presses, and value changes of a control.

Notify and filter events. LabVIEW categorizes user interface events into two different types of events:

- **Notify**—(Green arrow) Notify events inform you that a user action occurred. LabVIEW has already performed the default action associated with that event.
- **Filter**—(Red arrow) Filter events allow you to validate or change the event data before LabVIEW performs the default action associated with that event. You also can discard the event entirely to prevent the change from affecting the VI.



Activity 6-2: Lesson Review - Answers

1. Which of the following can NOT be used as the case selector input to a Case structure?
 - a. Error cluster
 - b. **Array**
 - c. Enum
 - d. String

2. How many events can an Event structure handle each time it executes?
 - a. As many as have occurred since the last time the event structure executed
 - b. One per configured event case
 - c. **One**

3. Which statements about event-driven programming versus polling are true?
 - a. **Events execute on demand.**
 - b. **Event-driven programming is less CPU-intensive.**
 - c. **Event structures handle all events in the order they occur.**
 - d. **Polling may fail to detect a change.**

7. Modularity.

Modularity. The degree to which a program is composed of discrete modules such that a change to one module has minimal impact on other modules. Modules in LabVIEW are called subVIs.

SubVI. A VI used within another VI. The degree to which a program is composed of discrete modules such that a change to one module has minimal impact on other modules. Modules in LabVIEW are called subVIs.

Icon. A VI icon is a graphical representation of a VI. It can contain text, images, or a combination of both. If you use a VI as a subVI, the icon identifies the subVI on the block diagram of the VI.

Connector Pane. A set of terminals that correspond to the controls and indicators of that VI, similar to the parameter list of a function call in text-based programming

languages. The connector pane defines the inputs and outputs you can wire to the VI so you can use it as a subVI.

Documentation. Create descriptions and tip strips for front panel objects in the properties dialog box for the object. Create descriptions for VIs on the Documentation page of the in the VI Properties dialog box.

- Descriptions appear in the Context Help window.
- Tip strips display when the cursor is over an object while the VI runs

Terminal settings. You can designate which inputs and outputs are required, recommended, and optional to prevent users from forgetting to wire subVI terminals. LabVIEW sets inputs and outputs of VIs you create to Recommended by default.

Appearance	Meaning	Description
Bold	Required	The block diagram containing the subVI will be broken if you do not wire the required inputs.
Plain	Recommended	The block diagram containing the subVI can execute if you do not wire the recommended or optional terminals. If you do not wire the terminals, the VI does not generate any warnings.
Dimmed	Optional	



Activity 7-1: Lesson Review - Answers

1. On a subVI, which terminal setting causes a broken VI if the terminal is not wired?
 - a. Required
 - b. Recommended
 - c. Optional

2. You must create a custom icon to use a VI as a subVI.
 - a. True
 - b. False

You do not need to create a custom icon to use a VI as a subVI, but it is highly recommended to increase the readability of your code.