EET 2259 Lab 7
Case Structures

OBJECTIVES
- Write LabVIEW programs using case structures.

Part 1. Summary of Loops and Case Structures
For Loops, While Loops, and Case Structures are three different kinds of structures. For Loops and While Loops are similar to each other. But case structures are very different from loops.

<table>
<thead>
<tr>
<th></th>
<th>For Loop</th>
<th>While Loop</th>
<th>Case Structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>When to Use</td>
<td>When you have an operation that you want to repeat a specified number of times.</td>
<td>When you have an operation that want to repeat until some condition becomes true or false. The condition could be a button being pressed by the user, a numeric variable being less than a certain value, etc.</td>
<td>When you want to choose one of several different operations depending on the value of some Boolean, numeric, or string variable.</td>
</tr>
<tr>
<td>Terminals</td>
<td>Count terminal that lets you set how many times to repeat the operation. Iteration terminal that keeps track of how many times the operation has been repeated.</td>
<td>Iteration terminal that keeps track of how many times the operation has been repeated. Conditional terminal that checks the condition at the end of each iteration to see whether to repeat the operation again.</td>
<td>Selector terminal to choose which of the different operations should be performed.</td>
</tr>
<tr>
<td>Terminal Wiring</td>
<td>Count terminal’s left-hand side is a numeric input that lets you set the number of times the loop will repeat. Count terminal’s right-hand side is a numeric output that you probably won’t use very often. Iteration terminal’s right-hand side is a numeric output that you can use to monitor the number of times the loop has repeated.</td>
<td>Iteration terminal’s right-hand side is a numeric output that you can use to monitor the number of times the loop has repeated. Conditional terminal’s left-hand side is a Boolean input that you should wire to the condition you want to check after each repetition to decide whether to repeat it again.</td>
<td>Selector terminal’s left-hand side is an input (can be Boolean, numeric, or string) that you should wire to the variable whose value determines which operation to perform.</td>
</tr>
</tbody>
</table>

Part 2. The Case Structure Compared to the Select Function
The case structure is similar to, but much more powerful than, the Select function.
- Recall that the Select function will choose between two inputs based the value of a Boolean input.
- The case structure will choose from among any number of different operations (not just two) based on the value of a Boolean, numeric, or string input.
In the first two programs below you’ll use a case structure to do things that you did in earlier labs using the Select function.

1. Create a VI whose front panel has two numeric controls labeled a and b, and one numeric indicator labeled Maximum. Using a case structure, wire the block diagram so that the numeric indicator displays the greater of the two numeric values. In other words, it displays a if \( a > b \), and it displays b if \( b > a \). (It can display either value if \( a = b \).) Your program should not contain any Select functions, and should also not use the function named Max & Min.

2. Modify your program so that it runs until the user presses a STOP button on the front panel. Save this VI as Lab7Maximum.vi, and show me your working program.

Let’s make some noise with LabVIEW!

1. Create a new VI whose front panel has a horizontal toggle switch labeled Coming or Going? and a string indicator labeled What to say. Using a case structure, wire the block diagram so that the string indicator displays the word “Hello” if the switch is switched off, and displays the word “Goodbye” if the switch is switched on. Your program should not contain any Select functions.

2. Using Beep.vi, which you’ll find on the Programming > Graphics and Sounds palette, modify your block diagram so that when the program displays the word “Hello” it also beeps the speaker with a 1000 Hz beep that lasts for one-half second. The program should not make any sound when the word “Goodbye” is displayed. Save this VI as Lab7ChooseNoisyGreeting.vi, and show me your working program.

Part 3. More Practice with Case Structures

In previous labs you’ve seen how to use the DAQ Assistant and myDAQ to perform analog input, analog output, digital input, and digital output. Now let’s write a program that lets the user choose which one of these tasks to perform.

1. Create a new VI whose front panel contains the following objects:
   - A dial labeled DAQ Function
   - A numeric indicator labeled Input Voltage
   - A knob labeled Output Voltage
   - An LED labeled Switch Status
   - A push button labeled LED Control
   - A STOP button

2. Configure the dial so that it can only be set to integer values. Also configure it to use text labels instead of numeric labels, and give it these four labels, in the order listed here:
   - Analog Input
   - Analog Output
   - Digital Input
   - Digital Output

---

EET 2259 Lab 7 – Page 2

Revised 12/18/2017
3. Using a single case structure, wire the block diagram so that LabVIEW performs whichever DAQ operation the user selects via the dial. In particular:
   - If the user selects **Analog Input**, then the program measures the voltage being produced by the trainer’s power supply and displays this voltage in the front panel’s numeric indicator.
   - If the user selects **Analog Output**, then the program produces a voltage equal to the setting of the front panel’s knob. This voltage should appear on the myDAQ’s channel 0 analog output terminal.
   - If the user selects **Digital Input**, then the program causes the front panel’s LED to light up or stay dark, depending on the position of the trainer’s Data Switch SW0.
   - If the user selects **Digital Output**, then the program causes the trainer’s LED 7 to light up or stay dark, depending on the position of the front panel’s push button.

4. Modify your program so that it continues to run until the user presses a STOP button on the front panel. Save this VI as Lab7DAQSelect.vi, and show me your working program.

In Lab #4 you wrote a program that converted resistor color codes to resistor values. Now let’s go in the other direction, converting resistor values to color codes.

1. Create a new VI whose front panel has a numeric control labeled **Number** and a string indicator labeled **Color Band**. Using a case structure, wire the block diagram so that the string indicator displays the color corresponding to the single digit entered by the user in the numeric control. If the user enters any number other than the integers from 0 through 9, the string indicator should display “Not a valid digit.”

2. Modify the program’s front panel by changing the numeric control’s label to **Resistor Value** and changing the string indicator’s label to **Color Band 1**. Then add two more string indicators labeled **Color Band 2** and **Color Band 3**. Wire the block diagram so that when the user runs the program with a valid resistor value (up to 10,000,000) entered into the numeric control, the string indicators display the correct colors. For example, if the user enters 680 for the resistor value, the string indicators should display blue-grey-brown. (Hint #1: You’ll probably want to use three separate case structures. Or you could create a subVI that contains a case structure, and use it three times.) (Hint #2: You’ll probably want to convert the resistor value from a number to a string, using the **Number To Decimal String** function. Then you can manipulate the decimal string with the **String Subset** function, which lets you pull out one or more individual characters from a string. For instance, if the user enters 2700, when you convert this to a decimal string you’ll have the string “2700,” and then you can use the String Subset function to pull off the first character (“2”), and the second character (“7”).)

3. Add a STOP button to the front panel and modify your block diagram so that the program continues to run until the user presses the STOP button. Save this VI as Lab7NumbersToColors.vi, and show me your working program.
Part 4. Structures Inside of Structures

Loops by themselves are powerful. Case structures by themselves are powerful. So combinations of loops and case structures are really powerful! Here are some of the possibilities:

- You can put one loop inside another loop. (You did this in Lab 5.)
- You can put a case structure inside a loop. (You’ve already done this in the previous programs in this Lab.)
- You can put one case structure inside another case structure.
- You can put a loop inside a case structure.

Each of these combinations can be useful under the right circumstances. The following exercises explore some of the possibilities.

1. Create a new VI whose front panel has a string indicator labeled Type of Person and two horizontal toggle switches labeled Male or Female? and Child or Adult? Without using any Select functions, wire the block diagram so that:
   - the indicator displays “It’s a boy!” if the user selects Male and Child;
   - the indicator displays “It’s a girl!” if the user selects Female and Child;
   - the indicator displays “It’s a woman!” if the user selects Female and Adult;
   - the indicator displays “It’s a man!” if the user selects Male and Adult.
   (Hint: Try using one or more case structures inside another case structure.)

2. Add a STOP button to the front panel and modify your block diagram so that the program continues to run until the user presses the STOP button. Save this VI as Lab7AgeSex.vi, and show me your working program.
Next, let’s make LabVIEW generate random strings of letters until they spell “OHIO.”

1. This first step requires one case structure and no loops. Create a new VI whose front panel has a dial labeled **Input Number** and a string indicator labeled **Output String**. Configure the dial so that it can only be set to the integers 0, 1, and 2. Wire the block diagram so that if the user sets the dial to 0, the letter “O” is displayed in the string indicator. If the user sets the dial to 1, the letter “I” is displayed in the string indicator. If the user sets the dial to 2, the letter “H” is displayed in the string indicator.

2. Next, modify this VI’s front panel by deleting the dial. Rewire the block diagram so that the program randomly generates an “O,” “I,” or “H” at the rate of one letter per second until the user presses a STOP button.

3. Modify this VI by duplicating your random generator and case structure so that now the program generates two letters every second. Concatenate these two letters and display them as a two-letter string. For instance, some of the two-letter strings that can appear are “HO,” “OI,” and “HI.” As above, the user should be able to press a STOP button to end the program.

4. Extending the same strategy, modify this VI so that it randomly generates four letters every second and displays them as a four-letter string.

5. Modify this VI so that the program stops either when the user presses the STOP button or when the word “OHIO” appears in the string indicator. Also, speed up the loop so that it executes ten times per second instead of once per second.

6. Modify this VI by adding a numeric indicator that is continually updated to show how many four-letter strings the program has generated. Save this VI as **Lab7CaseOhio.vi**, and show me your working program.

Finally, you’ll create a program that behaves almost the same as the previous one, but this time you’ll use a shift register on the loop instead of having four copies of the code that generates random letters.

1. Start by saving a new copy of the program under the name **Lab7ShiftOhio.vi**. Then delete three of the copies of the random number generators and case structures.

2. Add a shift register to your loop. Using the shift register, display the string made up of the last four random letters that have been generated. Be sure to initialize your shift register so that it is empty when the program starts. As in the previous program, the program should end when the user presses the STOP button or the word “OHIO” appears in the string indicator. Save this VI as **Lab7ShiftOhio.vi**, and show me your working program.

**Part 5. Review Questions**

As you may have realized, different types of data (numeric, Boolean, string, and so on) can be wired to a case structure’s selector terminal.
1. Of the programs listed below, which ones have numeric data wired to a case structure’s selector terminal? Place check marks in the blanks to indicate your answer.
   _____ Lab7Maximum   _____ Lab7ChooseNoisyGreeting
   _____ Lab7DAQSelect   _____ Lab7NumbersToColors
   _____ Lab7AgeSex   _____ Lab7CaseOhio
   _____ Lab7ShiftOhio

2. Of the programs listed below, which ones have Boolean data wired to a case structure’s selector terminal?
   _____ Lab7Maximum   _____ Lab7ChooseNoisyGreeting
   _____ Lab7DAQSelect   _____ Lab7NumbersToColors
   _____ Lab7AgeSex   _____ Lab7CaseOhio
   _____ Lab7ShiftOhio

3. Of the programs listed below, which ones have string data wired to a case structure’s selector terminal?
   _____ Lab7Maximum   _____ Lab7ChooseNoisyGreeting
   _____ Lab7DAQSelect   _____ Lab7NumbersToColors
   _____ Lab7AgeSex   _____ Lab7CaseOhio
   _____ Lab7ShiftOhio

*** This lab had 7 named programs for me to check. If you didn’t finish all of these during class, finish them after class. Then upload all 7 programs, along with any related subVIs, to the website by the due date. Also turn in your lab sheets at the beginning of class.****