# LabVIEW Loop – Shift Register

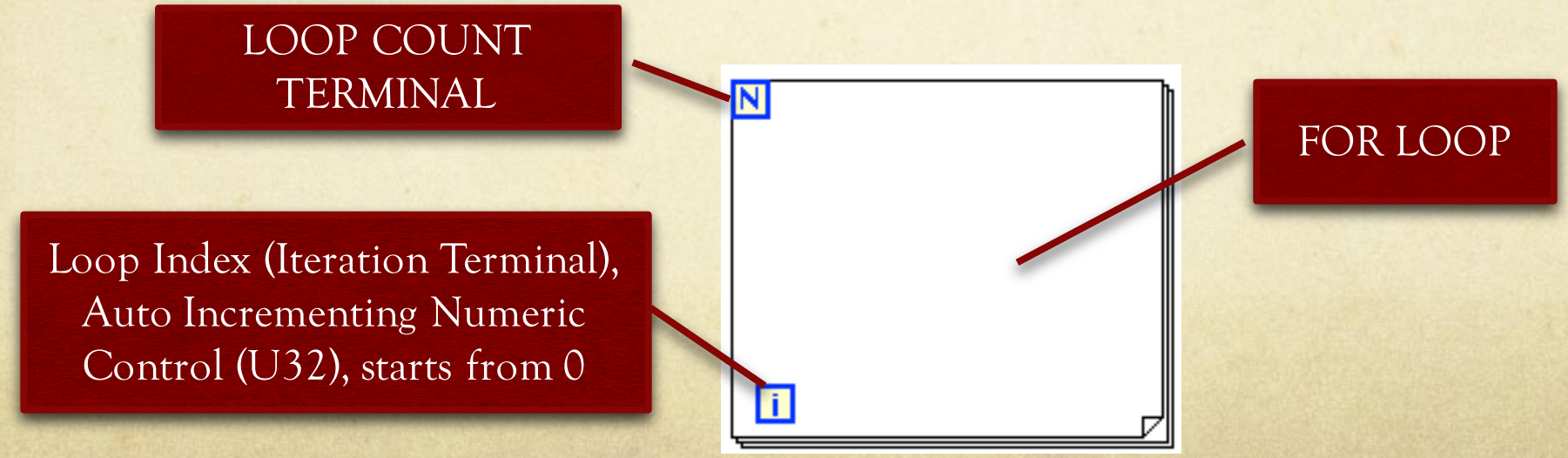
ABE 4423 /6423 – Bioinstrumentation II

Dr. Filip To

Ag and Bio Engineering, Mississippi State University

# FOR Loop and WHILE Loop

- LabVIEW has only two Loop structures: **FOR Loop** and **WHILE Loop**.
- FOR loop executes the code (sub-diagram) inside its border for a total of N times, where **N** is the value wired to its Loop Count terminal. The Loop Count is set before the FOR Loop is entered (it is wired from outside the loop). If 0 is wired to the Count terminal then the loop does not execute.



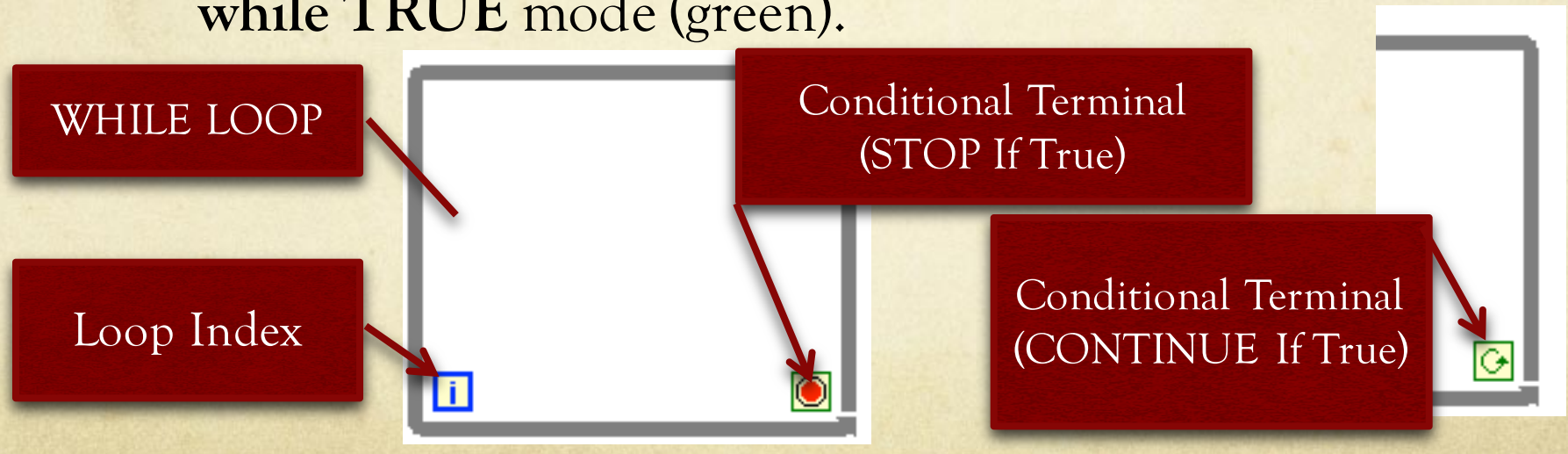
# FOR Loop

- The **Iteration** Terminal (the blue square with **i** in the middle) contains the number of iterations that have been completed. It is 0 at the start of the first iteration, 1 at the start of the second iteration, it increments from 0 to N-1 (N is the total number of iterations the loop executes before exiting).

- In text based programming, the FOR loop structure is equivalent to:
  *FOR I = 0 to N-1*
  *execute sub-diagram.*

# WHILE Loop

- **WHILE** loop executes the sub-diagram inside its border until the **Boolean value** wired to its **Conditional Stop Terminal** is TRUE.
- There are two modes of Conditional Terminal inside a WHILE loop: **Stop If TRUE** (red), and **Continue while TRUE** mode (green).

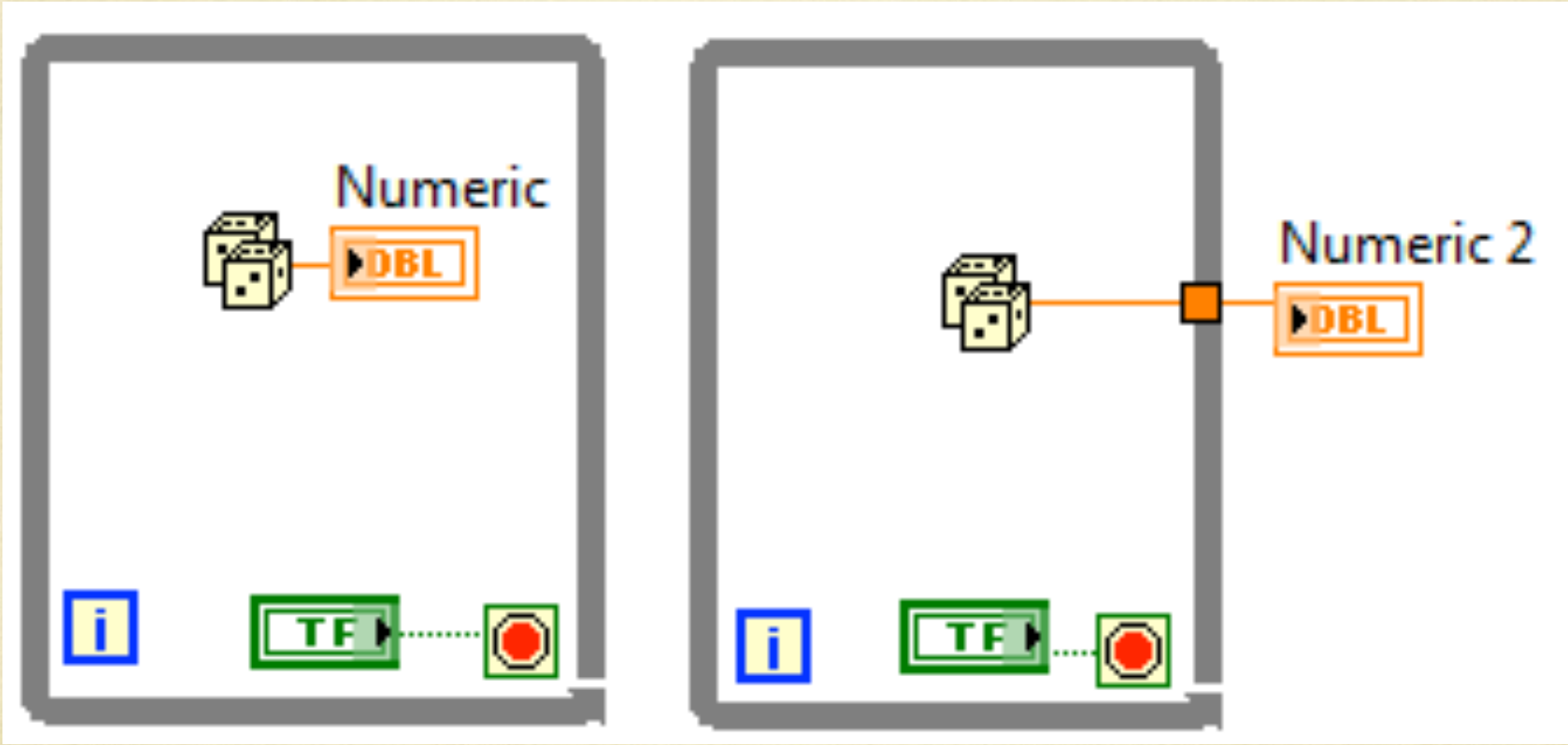


# WHILE Loop

- The mode of the Conditional Terminal of WHILE Loop is changed by using the POP-UP (right-click) on it, or by using the "Finger" cursor to click on it (you can make it Stop If True/ Continue While True) .

- Common behavior of Data inside a loop structure:
  - if a data is wired into a loop structure, that data is **captured** at entry of the loop. Only the value when the loop is entered is captured. Any changes to that data while the loop is executing is not detectable ("visible") inside the loop.
  - If a data is wired from inside the loop to an object outside a loop structure, that object will only get that data after the loop has exited (it gets the last data prior to loop exit).
  - If you want data (input data / output data) to be captured/updated at each iteration of a loop you must put that data object inside the loop structure.

# What's The Difference?



# Shift Registers

- It is used with a loop structure. It is attached to a loop structure by right-clicking the border of the loop and select **ADD SHIFT REGISTER.** It is a data holder used to capture data value during loop iterations.

- When you add a new shift register onto a loop, LabVIEW automatically add one pair: a **Shift-In** register (on left frame of the loop structure) and a **Shift-Out** register (on right frame of the loop structure)

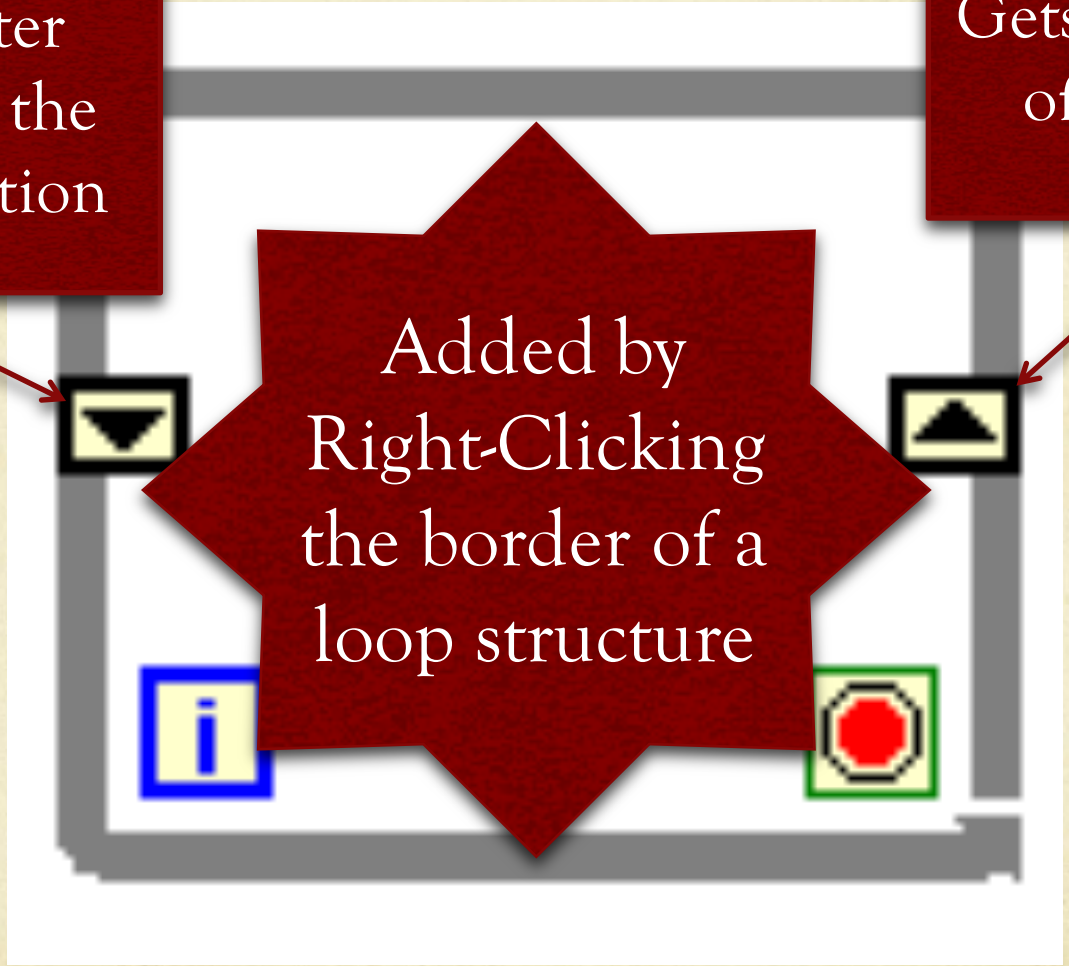
- **Shift-in** register captures its data at the beginning of an iteration, and **Shift-out** register captures its data at the end of an iteration.

- Shift-in register initially (at entry) gets its data from outside the loop and after that its data comes from shift-out register at the beginning of every new iteration. Shift-out register gets its data from the program inside the loop structure.

# Shift Registers

**Shift-in** Register
Gets its data at the start of an iteration

**Shift-Out** Register
Gets its data at end of an iteration



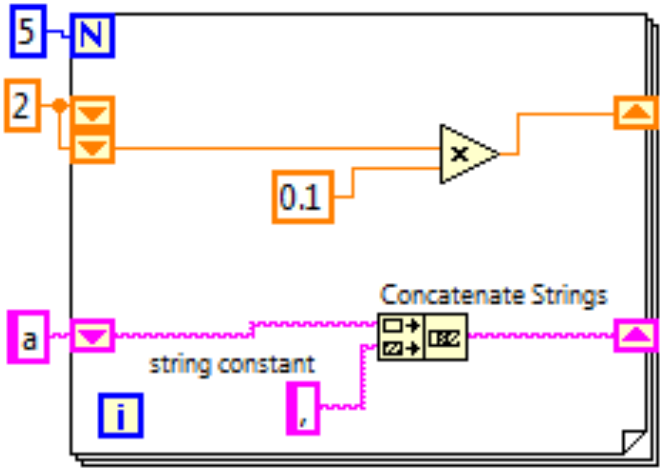
# Shift Registers

- Once a pair of Shift Register is created, it is possible to add elements to any of them to make it a "deeper" shift register to make it "remember" values from previous iterations. It is also possible to remove elements from a shift register to make it "shallower" by right-clicking it. The least number of elements in a shift register is 1.
- A shift register's content is shifted forward (toward the direction of the arrow (triangle symbol)) once in every iteration of the loop. The front most (bottom most for shift-in, top most for shift-out) element of a shift register gets shifted out and its content is filled with the data from the register behind it successively, hence the term Shift Register.
- There is no limit on how many pairs of shift registers that can be added to a loop structure or how deep a shift register you can make.

# How Shift Registers Work

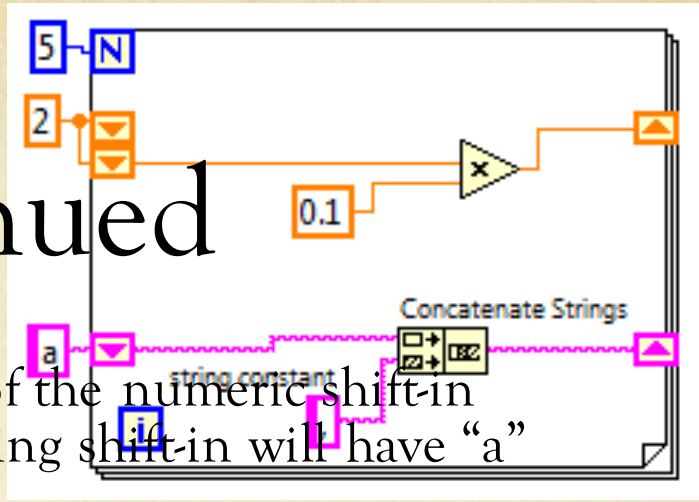
- Shift registers are **Polymorphic** (they can handle different data types), but shift-in and shift-out registers of the same group must have the same data type.
- When shift registers are added onto a loop structure, the **shift-in** register must be wired with an initial value/data (this is the value/data it will have upon entry into the loop).
- Inside the loop the **shift-out** must be wired with the same **data type** as the **shift-in** register.
- During iteration of the loop, data from the **shift-in** register is valid at start of iteration, and upon completion of an iteration the data of the font most of the **shift-out** register is moved into the **tail** position of the **shift-in** while the remainder data in the front positions of the shift registers get shifted forward one position. The process of shifting data forward and from shift-out to shift-in continues until the loop exits.

# In-class Examp




- See the following For Loop:
- There are two groups of shift Registers: the top group has two shift-in elements and one shift-out element
  The bottom group has one shift-in and one shift-out elements
- The shift-in elements of the top group is initialized to 2, and the bottom group is initialized to "a"
- Inside the loop: the numeric data is multiplied by 0.1, and the string data is concatenated with a comma.
- The For loop is set to run 5 iterations.

# Example: Continued




- Upon entry into the For Loop both elements of the numeric shift-in register will have 2 in them already, and the string shift-in will have “a”
- Iteration 1: the result of multiplication is 2*0.1 = 0.2, and the result of string operation is “a,”. At the end of iteration 1 the numeric shift-out register will have 0.2 and the string shift-out will have “a,”
- Iteration 2: The front element (bottom most) of the numeric shift-in will have 2 and the tail (top most) element will have 0.2, while the string sift-in will have “a,”. Then the code in the loop is executed again, which will result in 0.2 being put in the numeric shift-out, and “a,,” is put in the string shift-out.
- Iteration 3: repeat as described in iteration 2: the front element of numeric shift-in will have 0.2 and the tail element will have 0.02; the string shift-in will have “a,,”
- When the loop exits, the shift-out of string will have “a,,,,,” and the numeric shift-out will have 0.002

# In-Class Example

- Write a VI so that it can run continuously when the RUN button is clicked (without clicking the Run Continuously button) and the program will stop running when a STOP button is pressed or when the task is done.

- Make the program do the following approximation task: Given a formula $y = -x^3 + 3x^2 - 5x + 2$, find x that will make y = 0.5. It is easier to do this by approximation…

- Using a loop structure we iterate the value of x starting rom 0 and incrementing it a small amount in every loop iteration until the answer either matches the given y value of get close to it within a specified tolerance (say, 5%).

- A While loop is suitable for doing this because we don't know how many times we have to iterate. Case Structure is needed because there is a decision to be made, the decision to exit the loop when the answer is found or when the user press the STOP button.

# Homework Due Next Class Meeting

- Write a VI that estimate the height (h) of a partially filled sphere and the percent difference between estimated and given values (how do you define % difference?). The given values are the volume of the liquid and the radius of the sphere. The picture can be obtained from the web.
  The formula for computing the volume of a partially filled sphere is as follows: $Vp = \pi * (h^2 * r - h^3/3)$, where Vp is the volume of the liquid, h is the height of the liquid from the bottom of the sphere, and r is the radius of the sphere. Your program must also handle exceptions like $h > 2r$, $h < 0$, and $r < 0$ by displaying an appropriate pop-up message. Include Your name and an illustration (picture) of the shape (sphere) and the parameters. Include a numeric indicator showing how many iteration it takes to arrive at the correct result.

- Hint: Iterative approach in finding solution for this problem is done by approximation as follows: you create a **Tolerance** value below which you will accept the solution as correct. You create and **Increment Constant** which you will use to increment the estimated value of h after each iteration. You estimate the value of h by initially make it = 0 and use this value to compute a new volume using the formula and compare the result against the given volume. If the difference between the new v and the Given v is < Tolerance then the estimated h is the h value you sought, which you can display it and exit the program. Otherwise you will repeat the computation using a new h by incrementing its old value by the Increment Constant.