CIS1400 – Programming Logic and Technique

Topic 5 → Repetition Control Structures

Chapter Topics

5.1 Introduction to Repetition Structures

5.2 Condition-Controlled Loops: While, Do-While, and Do-Until

5.3 Count-Controlled Loops and the For Statement

5.4 Calculating a Running Total

5.5 Sentinels

5.6 Nested Loops

Visual Basic Program Examples

5.1 Introduction to Repetition Structures

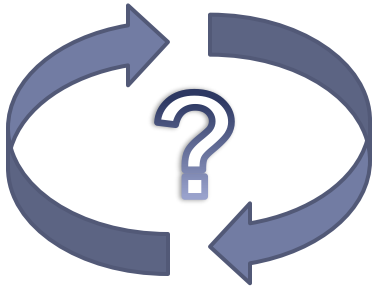
A repetition structure causes a statement or set of statements to execute repeatedly

Allows programmer to avoid problems associated with duplicate code

- Makes a program unnecessarily large
  - Can affect problem comprehension
  - Takes extra space on computer
- Is time consuming to write
  - Masks purpose of code sections that could reveal similarities
- Requires corrections or changes to be done many times
  - Increases maintenance efforts

5.2 Condition-Controlled Loops

- A condition-controlled loop iterates based upon the value of an evaluated condition
- Sometimes referred to as indeterminate loops.
- Types:
  - While Loop
    - While a condition is true, do some task
  - Do-While Loop
    - Do some task, while condition is true
  - Do-Until Loop
    - Do some task, while a condition is false (or until it's true)



- With all loops, be careful not to create infinite loops – always provide a way to break out of the loop!

5.2 Condition-Controlled Loops

The While Loop – pretest loop

- Loop iterates while a condition is true
- condition tested before iteration
- possible to have no iteration
- condition should be initialized/set before condition testing

While condition
Statement
Statement
End While

Body of loop is repeated while condition is true

Note alignment on pseudocode and symbols on flowchart

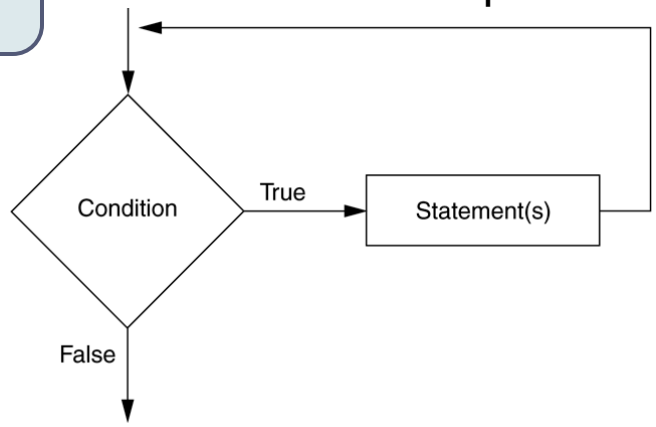
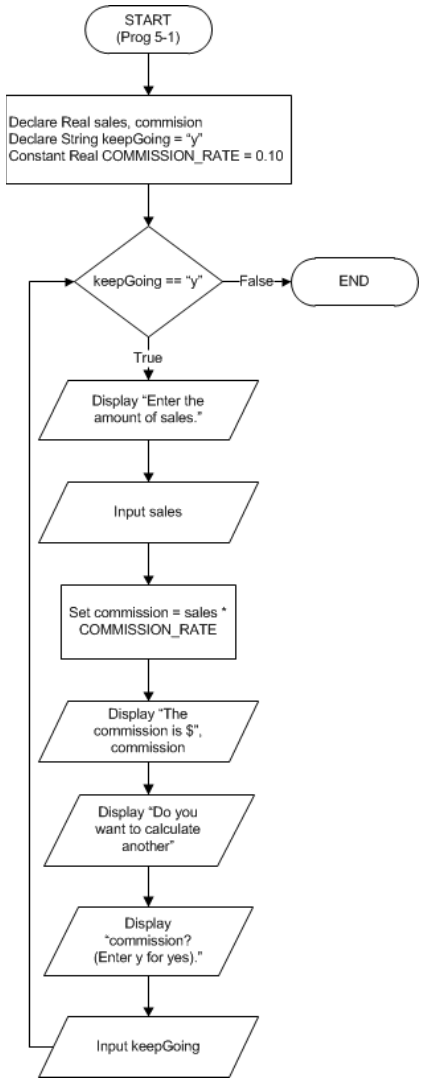


Figure 5-1 The logic of a `While` loop

While Loop Example – Program 5-1

```
// Variable declarations
Declare Real sales, commission
Declare String keepGoing = “y”
// Constant for the commission rate
Constant Real COMMISSION_RATE = 0.10
While keepGoing == “y”
  // Get the amount of sales
  Display “Enter the amount of sales.”
  Input sales
  // Calculate the commission
  Set commission = sales * COMMISSION_RATE
  // Display the commission
  Display “The commission is $”, commission
  Display “Do you want to calculate another”
  Display “commission? (Enter y for yes).”
  Input keepGoing
End While
```



While Loop Example – Program 5-3

Infinite Loops

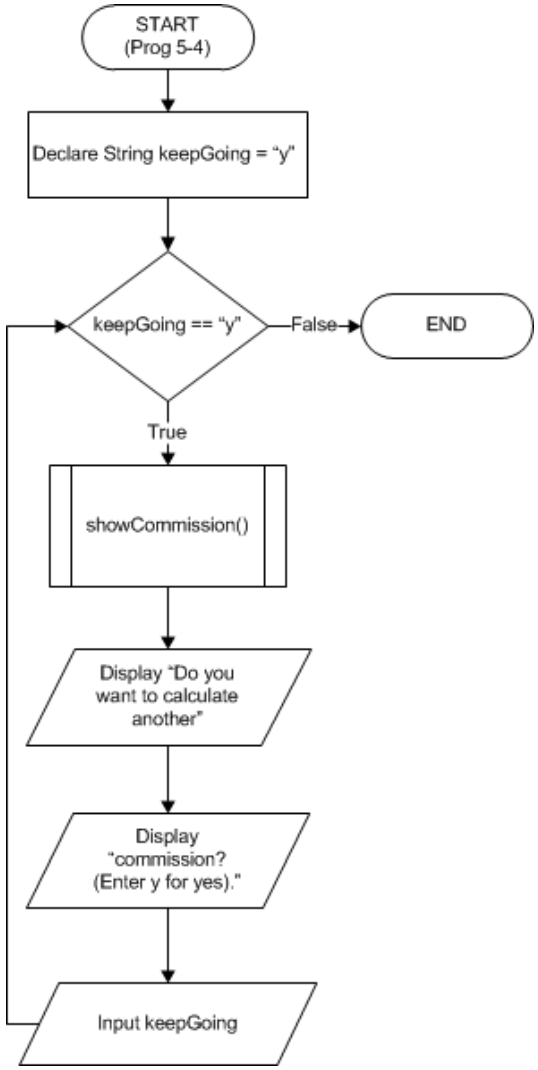
- Loop that has no way of stopping
  - Continues until program is interrupted
  - Occurs when code to modify condition is missing
- Loops must contain way to terminate
  - Rare exceptions

```
// Variable declarations
Declare Real sales, commission
Declare String keepGoing = “y”
// Constant for the commission rate
Constant Real COMMISSION_RATE = 0.10
// Warning! Infinite loop!
While keepGoing == “y”
   // Get the amount of sales
   Display “Enter the amount of sales.”
   Input sales
   // Calculate the commission
   Set commission = sales * COMMISSION_RATE
   // Display the commission
   Display “The commission is $”, commission
End While
```

While Loop Example – Program 5-4

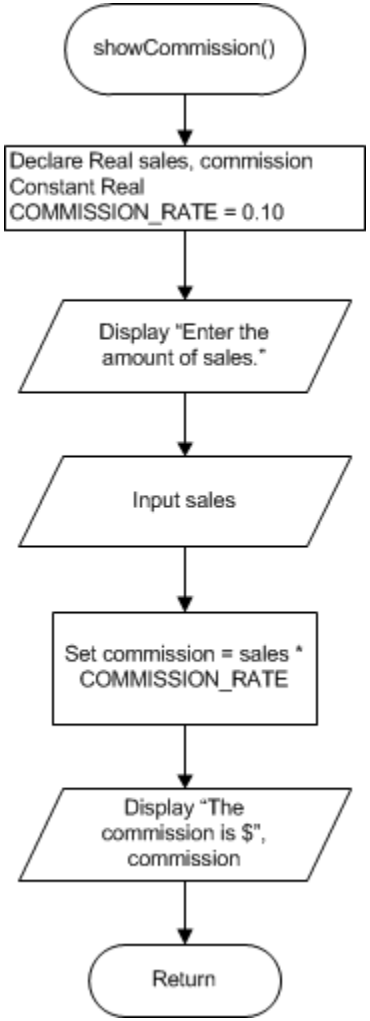
Could also prompt user for initial value.

```
Module main()
  // Local variable
  Declare String keepGoing = “y”
  // Calculate as many commissions
  // as needed.
  While keepGoing == “y”
    // Display a salesperson’s commission
    Call showCommission()
    // Do it again?
    Display “Do you want to calculate another”
    Display “commission? (Enter y for yes).”
    Input keepGoing
  End While
End Module
```



While Loop Example – Program 5-4

```
// The showCommission module gets the
// amount of sales and displays the
// commission.
Module showCommission()
  // Local variables
  Declare Real sales, commission
  // Constant for the commission rate
  Constant Real COMMISSION_RATE = 0.10
  // Get the amount of sales
  Display “Enter the amount of sales.”
  Input sales
  // Calculate the commission.
  Set commission = sales * COMMISSION_RATE
  // Display the commission
  Display “The commission is $”, commission
End Module
```



5.2 Condition-Controlled Loops

The Do-While Loop – posttest loop

- Loop iterates while a condition is true
- condition tested after iteration
- must have at least one iteration
- condition should be reset within loop body

Do
Statement
Statement
While condition

Body of loop is repeated while condition is true

Note alignment on pseudocode and symbols on flowchart

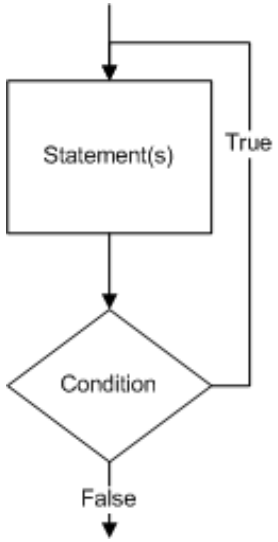
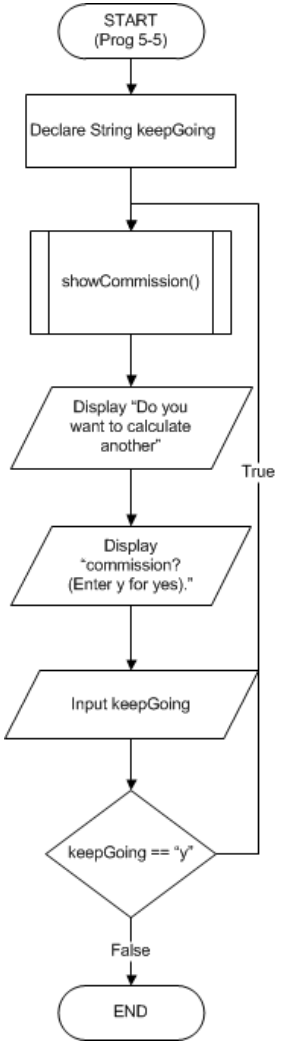


Figure 5-7 The Logic of a Do-While loop

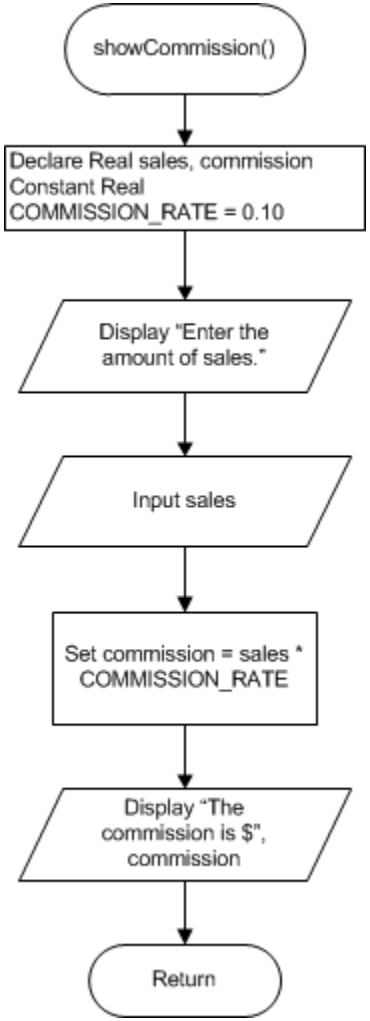
Do-While Loop Example –Program 5-5

```
Module main()
  // Local variable
  Declare String keepGoing
  // Calculate commissions as many
  // times as needed.
  Do
    // Display a salesperson's commission.
    Call showCommission()
    // Do it again?
    Display "Do you want to calculate another"
    Display "commission? (Enter y for yes)."
    Input keepGoing
  While keepGoing == "y"
End Module
```



Do-While Loop Example –Program 5-5

```
// The showCommission module gets the
// amount of sales and displays the
// commission.
Module showCommission()
  // Local variables
  Declare Real sales, commission
  // Constant for the commission rate
  Constant Real COMMISSION_RATE = 0.10
  // Get the amount of sales
  Display “Enter the amount of sales.”
  Input sales
  // Calculate the commission
  Set commission = sales * COMMISSION_RATE
  // Display the commission
  Display “The commission is $”, commission
End Module
```



5.2 Condition-Controlled Loops

The Do-Until Loop

- Loop iterates until a condition is true
  - not all languages support this type of loop
- condition tested after iteration
- must have at least one iteration
- condition should be reset within loop body

Do

Statement

Statement

Until condition

Body of loop is repeated until condition is true

Note alignment on pseudocode and symbols on flowchart

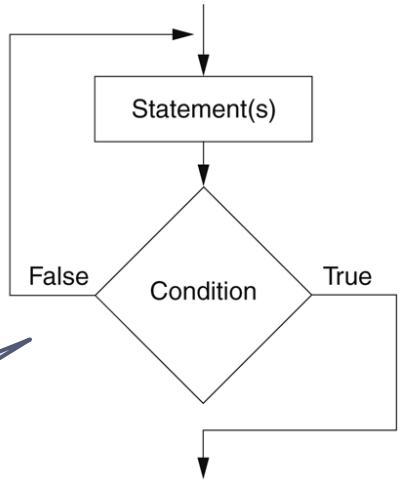
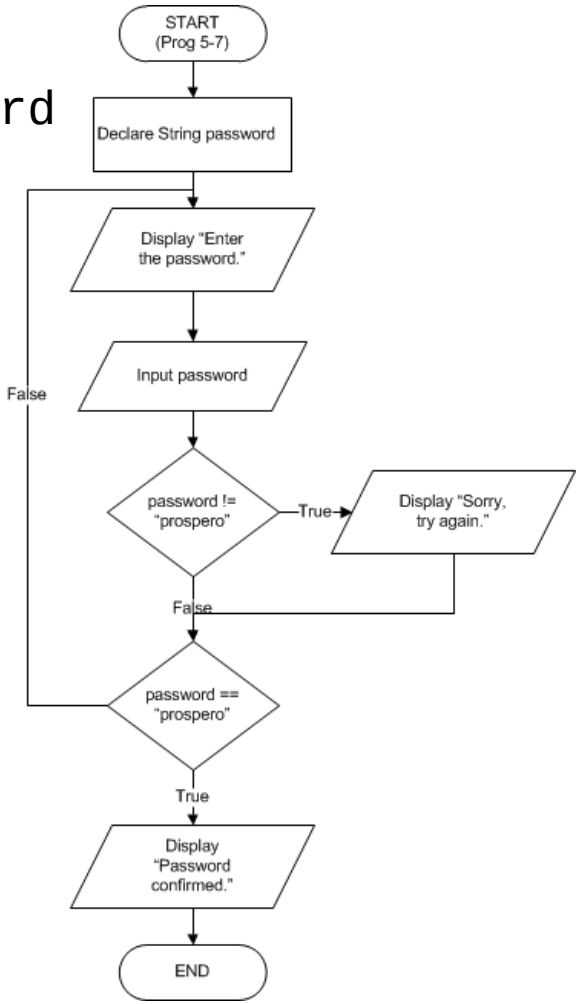


Figure 5-10 The logic of a `Do-Until` loop

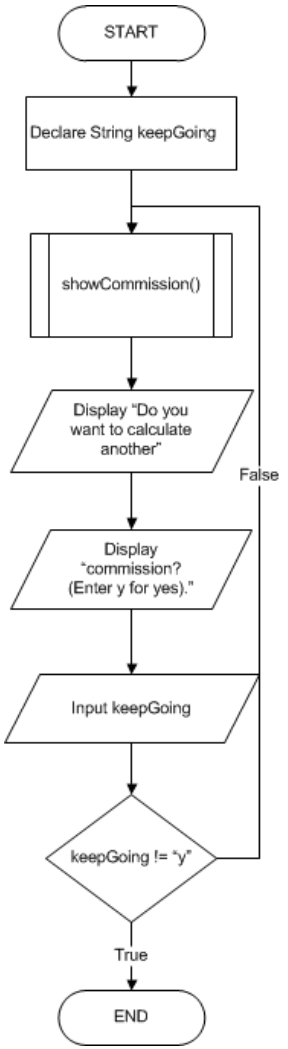
Do-Until Loop Example – Program 5-7

```
// Declare a variable to hold the password
Declare String password
// Repeatedly ask the user to enter a password
// until the correct one is entered
Do
  // Prompt the user to enter the
  // password
  Display “Enter the password.”
  Input password
  // Display an error message if the
  // wrong password was entered.
  If password != “prospero” Then
    Display “Sorry, try again.”
  End If
Until password == “prospero”
// Indicate that the password is
// confirmed
Display “Password confirmed.”
```



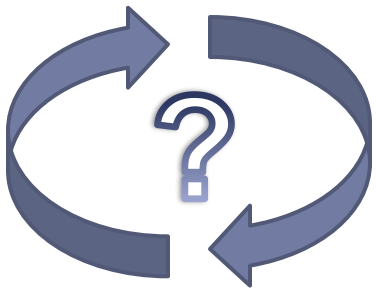
Do-Until Loop Example – Sales Commissions

```
Module main()
  // Local variable
  Declare String keepGoing
  // Calculate commissions as many
  // times as needed.
  Do
    // Display a salesperson's commission.
    Call showCommission()
    // Do it again?
    Display "Do you want to calculate another"
    Display "commission? (Enter y for yes)."
    Input keepGoing
  Until keepGoing != "y"
End Module
```



5.2 Condition-Controlled Loops

- Which loop to use?
  - While Loop
    - Repeat task as long as condition is true
    - Ideal when loop may not execute at all
      - Condition is false at start
  - Do-While Loop
    - Repeat task as long as condition is true
    - Ideal when loop should execute at least once
      - Condition is true or false at start
  - Do-Until Loop
    - Repeat task as long as condition is false
    - Ideal when loop should execute at least once
      - Best for task performed until condition is true



5.3 Count-Controlled Loops

- A count-controlled loop iterates a specific number of times
- Sometimes referred to as determinate loops
- A for loop is best used for this situation

For counterVariable = startingValue to maxValue

statement

statement

End For

- There is an Initialization, Test, and Increment expression that controls the loop

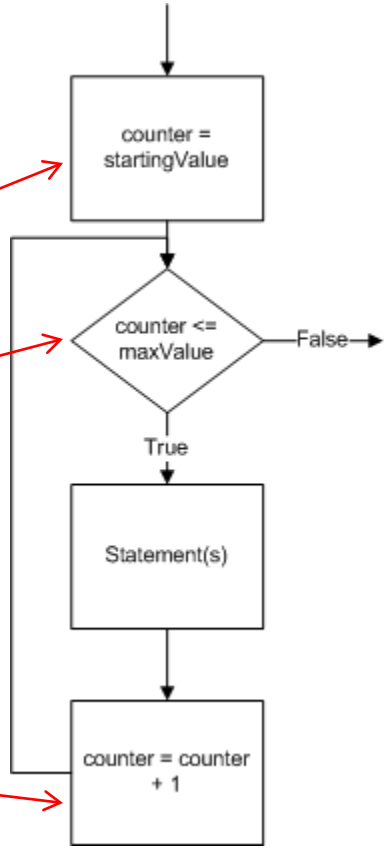
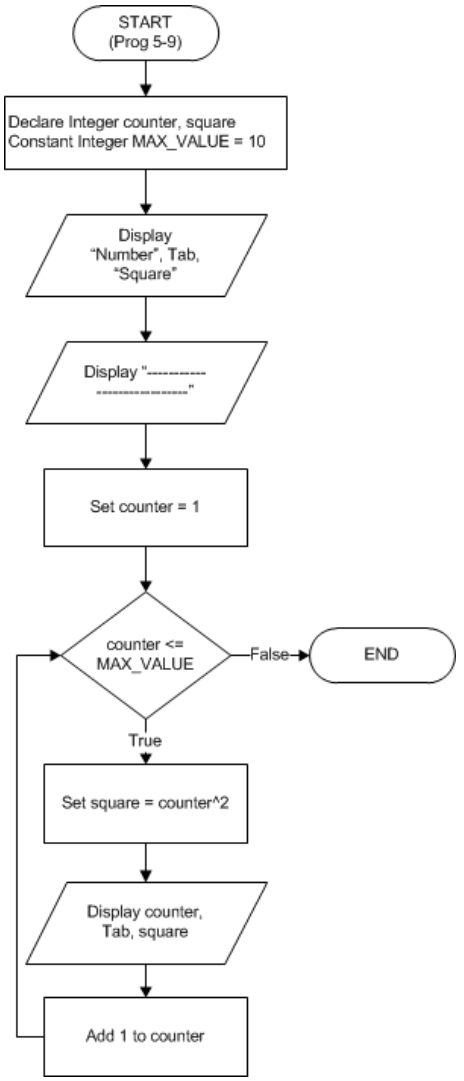


Figure 5-12 Logic of a count-controlled loop

For Loop Example – Program 5-9

```
// Variables
Declare Integer counter, square
// Constant for the maximum value
Constant Integer MAX_VALUE = 10
// Display table headings.
Display “Number”, Tab, “Square”
Display “-----------------------”
// Display the numbers 1 through 10 and
// their squares.
For counter = 1 to MAX_VALUE
  // Calculate number squared.
  Set square = counter^2
  // Display number and number squared.
  Display counter, Tab, square
End For
```



5.3 Count-Controlled Loops

For loops can also increment by more than one, count backwards by decrementing

For counterVariable = startingValue to endValue Step stepValue

statement

statement

End For

```
Declare Integer counter
For counter = 1 To 5 Step 1
  Display “Hello World”
End For
```

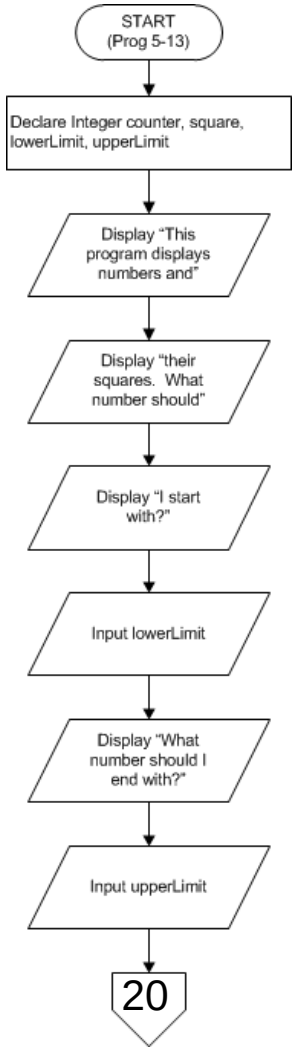
```
Declare Integer counter
For counter = 5 To 1 Step -1
  Display “Hello World”
End For
```

```
Declare Integer counter
For counter = 2 To 10 Step 2
  Display “Hello World”
End For
```

```
Declare Integer counter
For counter = 10 To 2 Step -2
  Display “Hello World”
End For
```

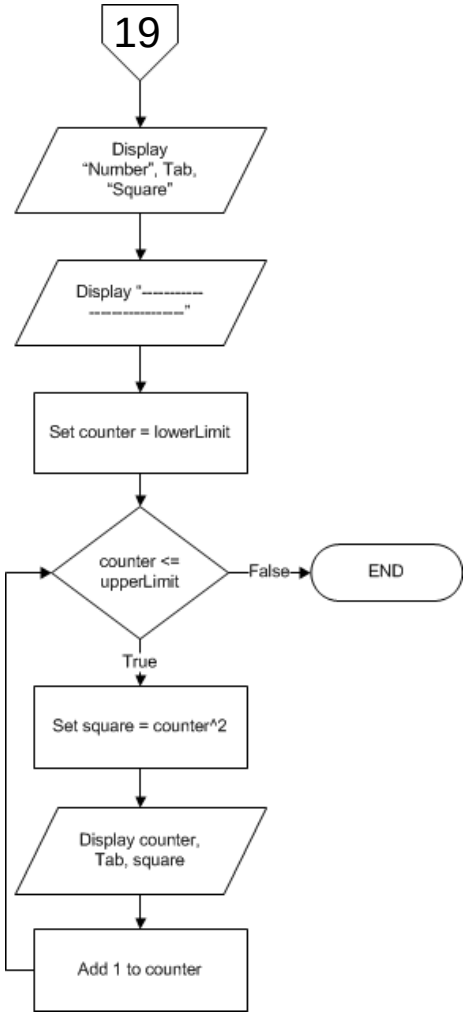
For Loop Example – Program 5-13

```
// Variables
Declare Integer counter, square,
  lowerLimit, upperLimit
// Get the lower limit
Display “This program displays numbers and”
Display “their squares.  What number should”
Display “I start with?”
Input lowerLimit
// Get the upper limit.
Display “What number should I end with?”
Input upperLimit
```



For Loop Example – Program 5-13

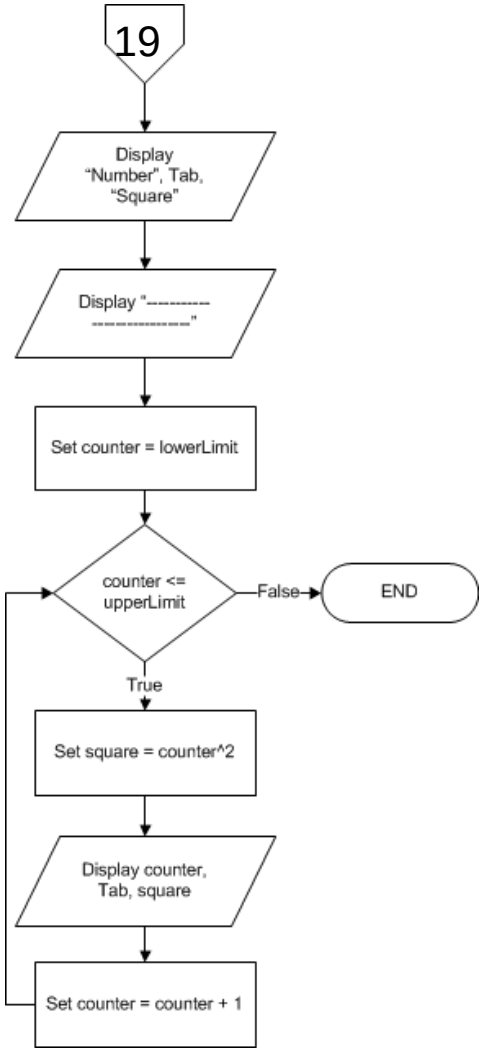
```
// Display table headings.
Display “Number”, Tab, “Square”
Display “------------------------”
// Display the numbers and their squares.
For counter = lowerLimit to upperLimit
  // Calculate number squared.
  Set square = counter^2
  // Display number and number squared.
  Display counter, Tab, square
End For
```



For Loop Example – Program 5-13

Creating a count-controlled While Loop of previous

```
// Display table headings.
Display “Number”, Tab, “Square”
Display “------------------------”
Set counter = lowerLimit
// Display the numbers and their squares.
While counter <= upperLimit
  // Calculate number squared.
  Set square = counter^2
  // Display number and number squared.
  Display counter, Tab, square
  Set counter = counter + 1
End While
```



5.3 Count-Controlled Loops

General loop concerns

- Do not forget to initialize the loop control variable
- Do not forget to modify the loop control variable
- Make sure loop is easily understood
- Many loops are interchangeable, but generally should keep in mind the following when selecting a loop type:
  - Number of iterations known?
    - Yes → use determinate loop (counter controlled)
    - No → use indeterminate loop (condition controlled)
  - Must have at least one iteration?
    - Yes → use posttest loop (do-while, do-until)
    - No → use pretest loop (while, for)

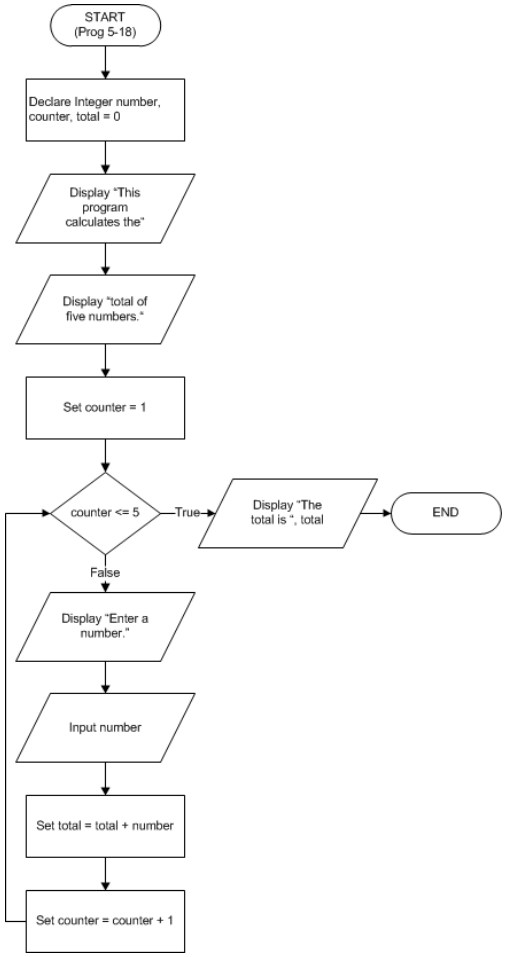
5.4 Calculating a Running Total

A running total (*accumulator*) is a sum of number that accumulates with each iteration of a loop

- Loop reads in each number
- Variable to accumulate numbers during each read

Program 5-18 Calculate a running total

```
Declare Integer number, counter, total = 0
Display “This program calculates the ”
Display “total of five numbers”
For counter = 1 to 5
  Display “Enter a number.”
  Input number
  Set total = total + number
End For
Display “The total is ”, total
```



5.5 Sentinels

Alternative designs for processing list of values:

- Ask user at end of each loop iteration if there is another value to process
  - Cumbersome for long lists
- Ask user at the beginning of the loop how many times the loop should process
  - Cumbersome for long lists and if user does not know number of items in list
- Ask user to input a number that would not be a valid value

A sentinel is a special value that marks the end of a list of values

- used as stop values for conditional loops
- Example: Program 5-19

5.6 Nested Loops

All loops can be nested:

- Loop inside of a loop
- No limitations on the types of loops that can be nested

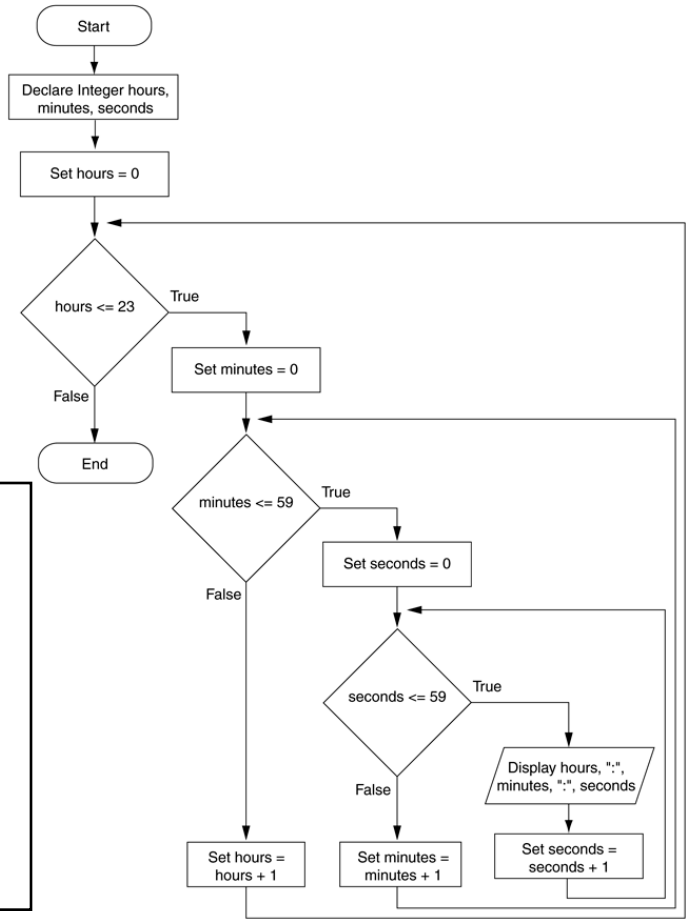


Figure 5-21 Flowchart for a clock simulator

```
Declare Integer hours, minutes, seconds
Display “Clock Simulator Program”
For hours = 0 to 23
  For minutes = 0 to 59
    For seconds = 0 to 59
      Display hours, “:”, minutes, “:”, seconds
    End For
  End For
End For
```