Learning Objectives

5.1 Introduction to Repetition Structures

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

Allow a programmer to avoid duplicate code

- Duplicate code makes a program large
- Write a long sequence of statements is time consuming
- If part of the duplicate code has to be corrected or changed, then the change has to be done many times

5.2 Condition-Controlled Loops (1 of 5)

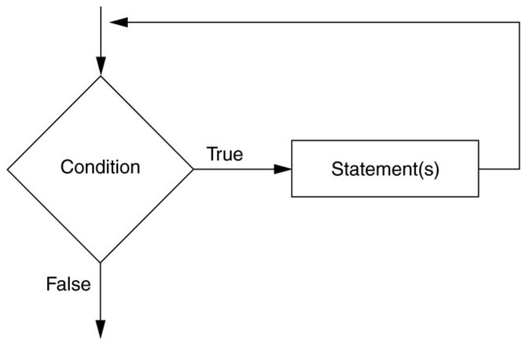
- While Loop
  - While a condition is true, do some task
- Do-While Loop
  - Do some task, while condition is true
- Do-Until Loop
  - Do some task, while a condition is false (or until it's true)
- With all loops, be careful not to create infinite loops – always provide a way to break out

5.2 Condition-Controlled Loops (2 of 5)

The While Loop – pretest loop

```
While condition
    Statement
    Statement
End While
```

Figure The logic of a `While` loop

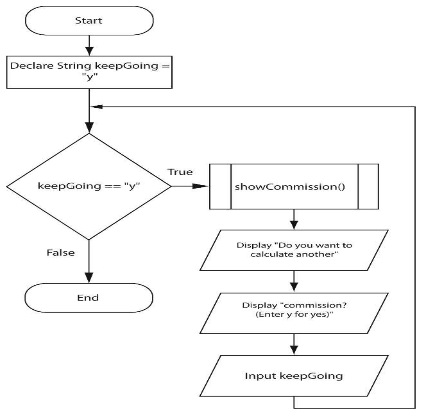


5.2 Condition-Controlled Loops (3 of 5)

Working with Modules and Loops

- To run a program multiple times, modules can be put within a loop

FigureThe main module of Program



Start
Declare String keepGoing = "y"
keepGoing == "y"
True
showCommission()
Display "Do you want to calculate another"
Display "commission? (Enter y for yes)"
Input keepGoing
False
End

5.2 Condition-Controlled Loops (4 of 5)

The Do-While Loop – posttest loop

```
Do
   Statement
   Statement
While condition
```

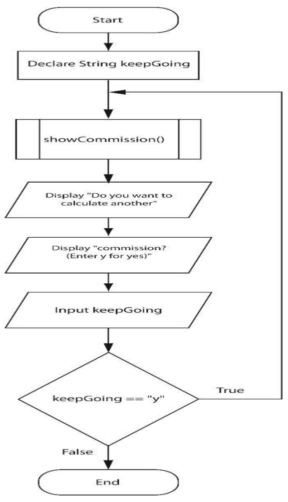


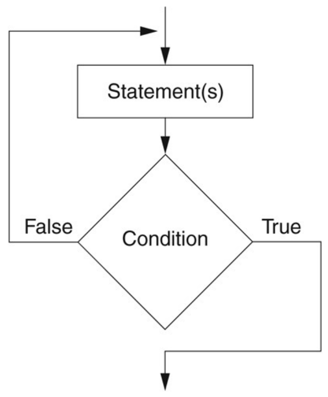
Figure Flowchart for the main module in Program

5.2 Condition-Controlled Loops (5 of 5)

The Do-Until Loop

- Loop iterates until a condition is true, but not all languages support this type of loop

Figure The logic of a `Do-Until` loop



5.3 Count-Controlled Loops (1 of 3)

A count-controlled loop iterates a specific number of times

- A for loop is best used for this situation

```
For counterVariable = startingValue to maxValue
        statement
        statement
End for
```

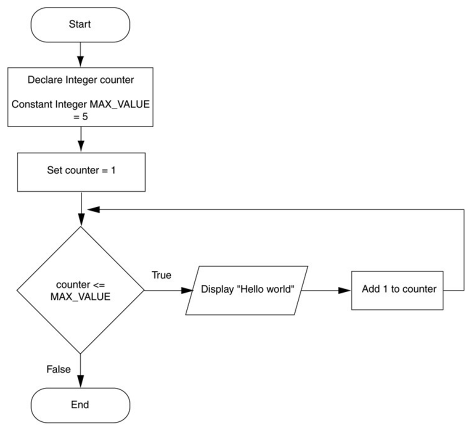
- There is an Initialization, Test, and Increment expression that controls the loop

5.3 Count-Controlled Loops (2 of 3)

For loops can also increment by more than one, count backwards by decrementing, or allow the user to control the number of interactions

The for loop in action

Figure Flowchart for Program



5.3 Count-Controlled Loops (3 of 3)

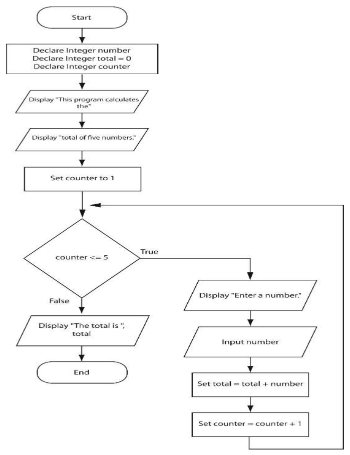
General loop concerns

- Do not forget to initialize the loop control variable
- Do not forget to modify the loop control variable
- Many loops are interchangeable, but generally
  - Use while loop when loop may not have to process
  - Use do while when it must process at least once
  - Use for loop with specific number of iterations

5.4 Calculating a Running Total

A running total is a sum of number that accumulates with each iteration of a loop

Figure Flowchart for Program


Start
Declare Integer number
Declare Integer total = 0
Declare Integer counter
Display "This program calculates the"
Display "total of five numbers."
Set counter to 1
counter <= 5
True
False
Display "Enter a number."
Input number
Set total = total + number
Set counter = counter + 1
Display "The total is ", total
End

5.5 Sentinels

A sentinel is a special value that marks the end of a list of values, used as stop values for loops

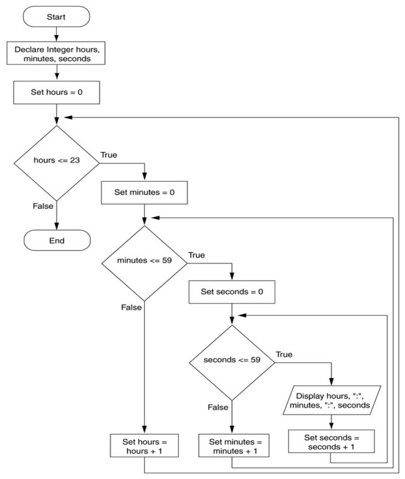
How it can be done

- Ask the user at the end of each loop iteration, if there is another value to process
- Ask the user at the beginning of the loop, how many times the loop should process

5.6 Nested Loops

All loops can be nested, that is, a loop inside of a loop

Figure Flowchart for a clock simulator


Start
Declare Integer hours, minutes, seconds
Set hours = 0
hours <= 23
True
False
End
Set minutes = 0
minutes <= 59
True
False
Set seconds = 0
seconds <= 59
True
False
Display hours, ":", minutes, ":", seconds
Set seconds = seconds + 1
Set minutes = minutes + 1
Set hours = hours + 1