Chapter 4

Control Structures: Part I

Algorithms

- Actions to be executed
  - order in which these actions are to be executed
- Pseudocode Example p. 96
  - artificial and informal language similar to English
  - statements are not executed on computer
  - used to determine flow of program and reduce logic errors
  - can be easily converted to programming language
  - used for executable statements

Introduction to Control Structures

- Execution of program commands are usually sequential
- Transfer of control- allows programmer to guide program execution
- Main control structures: sequence structures, selection structures, and repetition structures
- Flowchart- graphical representation of an algorithm or portion
  - drawn using special symbols

Introduction to Control Structures

- Useful for developing and representing algorithms (Figure 4.1, p. 92; Figure 4.2, p. 94)
- Rectangle- action symbol(any type of action)
- Oval- contains Begin, Start, or End
- Circle- used when represents only a portion of algorithm (connector symbol)
- Diamond- decision symbol
- VB has three types of selection structures: If/Then, If/Then/Else, and Select Case

Introduction to Control Structures

- If/Then- true makes selection or skips selection (single)
- If/Then/Else- true makes selection or false makes another selection (double)
- Select Case- multiple selections
- VB has six types of repetition structures: While, Do While, Do Until, Do Loop/While, Do Loop/Until, For/Next
- Keywords: If, Then, Else, While, Do, Until

Introduction to Control Structures

- Keywords- Loop, Select, Case, For, Next
- VB has ten control structures
- Single-entry/single-exit control structures make it easy to build programs (stacking)
- Control structures can be connected by nesting
- Note: any program can be written using ten control structures and combined in two ways

Introduction to Control Structures

- If/Then Selection Structure
  - pseudocode- If student,s grade is greater than or equal to 60 Display "Passed"
  - VB code: If grade >= 60 Then
  - lblStatus.Caption = "Passed"
  - End If
  - Flowchart: Figure 4.2, p. 94
  - Flowcharts for control structures contain only rectangle symbols indicate actions to be performed
  - Diamonds indicate decisions to be made

Introduction to Control Structures

- Example of action/decision mode of programming
- If/Then/Else Selection Structure:
  - If Student's grade is greater than or equal to 60
    - Display "Passed"
  - Else
    - Display "Failed"
- Pseudocode

Introduction to Control Structures

- if grade >= 60 then
  - lblStatus.Caption = "Passed"
- else
  - lblStatus.Caption = "Failed"
- end if
  - VB code
- Note1: compiler ignores whitespace characters
  - blanks, tabs, and newline (used for indention and vertical spacing)
- Note2: use uniform spacing conventions

Introduction to Control Structures

- Note3: spanning multiple lines requires End If
  - exception use of line-continuation character
- Function IIF has three arguments: condition, value returned when True, value returned when False
- lblStatus.Caption = IIF(grade >= 60, "Passed",
  - "Failed")
- Note4: performs same action as If/Then/Else
- Nested If/Then/Else allow multiple case tests

Introduction to Control Structures

- Function IIf has three arguments:
  - condition, value returned when True, value returned when False
  - lblStatus.Caption = IIf(grade >=60, "Passed", "Failed")
- Note1: performs same action as If/Then/Else
- Nested If/Then/Else allow multiple case tests
  - place one within another
  - Example coding on p. 96
- Note2: ElseIf could be used to simplify coding

Introduction to Control Structures

- Note3: ElseIf requires only on End If
- Place a blank line before and after every control structure makes easier identification
- Function Switch:
  - related to If/Then/Else
  - passed a condition or value
  - any number of condition-value pairs can be passed
  - condition True: value associated with condition returned
  - condition False: returns null

Introduction to Control Structures

- Note4: does not provide Else situation
  - value must be paired with conditions
- Example: lblStudentGrade.Caption = Switch
  - (grade >= 90, “A”, _
  - grade >= 80, “B”,_
  - grade >= 70, “C”,_
  - grade >= 60, “D”,_
  - grade < 60, “F”)
- Note5: similar to If/Then/Else logic
  - not providing condition causes run-time error

Introduction to Control Structures

- Note6: If/Then/Else is also referred to as block
- Body of control structure may contain one or more statements
- Example: If grade >= 60 The
- lblStatus.Caption = “Passed”
- Else
- lblStatus.Caption = “Failed”
- End If

Introduction to Control Structures

- While/Wend Repetition Structure
- repetition allows action to be repeated if True or False of some condition
- Common error:
  - not providing an action that will eventually cause a False condition (creates infinite loop)
- Example: Dim product As Integer
  - product = 2
  - While product <= 1000
  - product = product * 2
  - Wend

Introduction to Control Structures

- Ends when product = 1024
- can have more than one statement in loop
- Flowchart Figure 4.4, p. 99
- Do While/Loop Repetition Structure
  - Dim product As Integer
  - product = 2
  - Do While product <= 1000
  - product = product * 2
  - Loop
- Same logic as While/Wend

Introduction to Control Structures

- Do Until Loop Repetition Structure
- Test condition for False situation
- used when situation is better tested in negative situation
  - Example: Dim product As Integer
  - product = 2
  - Do Until product > 1000
  - product = product * 2
  - Loop
- Figure 4.6, p. 100

Introduction to Control Structures

- Formulating Algorithms: Case Study 1
- Counter-controlled
- Create program that displays in a label up to nine “#” characters (pseudocode p. 101)
- User inputs number (from 1-9) indicating number of “#” characters to display
- Interface: Figure 4.8, p. 101
- Interface specification: Figure 4.9, p. 102
- Code: Figure 4.10, p. 103

Introduction to Control Structures

- Formulating Algorithms with Top-down, Stepwise Refinement
- Sentinel-controlled repetition
- Develop class averaging program that will process an arbitrary number of grades each time program is run
- Questions:
  - How will program know when to calculate and display class average?

Introduction to Control Structures

  - How will program know when to stop input of grades?
- Solution- use special value called a sentinel value
  - signal, dummy, or flag to indicate end of input
  - also called indefinite repetition since number of reps is not know before hand
- Note: sentinel value cannot be a valid input value
  - Example: -1 would be a good sentinel value

Introduction to Control Structures

- Pseudocode: p. 106 (refined version)
- Interface: Figure 4.12, p. 107
- Specifications: Figure 4.13, p. 107-108
- Code: Figure 4.14, p. 108-109
- Total- variable used to accumulate sum of series of values
  - initialized to zero
- Counter- variable used to count iterations
  - initialized to zero

Introduction to Control Structures

- Note1: if an accumulator variable is not initialized to zero
  - could contain garbage values when created
- Single- handles floating-point numbers
  - ! is type declaration
- Note2: in sentinel-controlled loop
  - prompts requesting data entry should explicitly state sentinel value
- String-sequence of characters encoded in double quotes ($-type declaration)

Introduction to Control Structures

- Function InputBox used to input first grade
- Note3: integer division returns a whole number
- message = message & Format$(average, "Fixed")
  - concatenates message (Class average is) to formatted average (2 decimal places)
- Formulating Algorithms with Top-down, Stepwise Refinement
  - nested control structures

Introduction to Control Structures

- Problem: Write program that draws sequence of $ characters on form.
- The side of square (number of $ characters to be printed side by side) should be input by user and should be in range 1-12
- Pseudocode for top:
  - draw square of $ characters on form (complete program)
- 1st refinements:
  - initialize variables

Introduction to Control Structures

- Prompt for the side of square
- input side of square
- validate side is within prompt range
- print sequence

- Initialize variables:
  - side to the value input
  - row to one
  - column to one
- Input side of square (textbox or inputbox)

Introduction to Control Structures

- Validate input within range refined:
  - if side less than or equal to 12 then
    - if side is greater than 0 then
  - (nested control structures)
- Print square refinement:
  - while row is less than or equal to side
    - set column to one
    - While column is less than or equal to side
      - print $
      - increment column by one
      - print

Introduction to Control Structures

    - Increment row by one
- Beep statement- sounds a beep through the computer speaker
- Frequency and duration of beep is hardware and system dependent
- Note: avoid using more than three levels of nesting
- pseudocode Figure 4.15, p. 113
  - interface Figure 4.16, p.113 ;Specs:4.17, p.114 code:Figure 4.18, p.115