Chapter 3

Introduction to Visual Basic Programming

Visual Programming and Event Programming

- Visual Programming
  - create GUI's by pointing and clicking mouse
  - eliminates need for writing code for GUI's
- Programmer writes code to describe what happens when user interacts with interface (events)
- Events are passed on to windows operating system

Visual Programming and Event Programming

- Event-Driven programming creates code to represent events
  - user drives program, not programmer
    - user friendly
  - interface will remain in an event-monitoring state until user responds (user-driven)
  - Event procedures respond to events and are automatically generated by VB
  - programming code responds to certain events

A Simple Program : Printing a Line Of Text on the Form

- Create two buttons: Print and Exit
- CommandButton object is used to create two buttons
- Form, command buttons, and properties
  - Figure 3.2, p. 52
- Properties window contains Object box
  - determines which object's properties are displayed

A Simple Program : Printing a Line Of Text on the Form

- Properties:
  - TabIndex determines which control gets the focus
    - (active control) user uses tab key to move focus
    - range 0 to last control
    - Control with focus has darker border and dotted inner square on its face
    - 1st control added is assigned index of 0
      - can be changed in properties window
  - Coding Print button's event procedure

A Simple Program : Printing a Line Of Text on the Form

- Code window:
  - Private sub cmdDisplay_Click( )
    - procedure definition header
  - View:
    - Procedure View displays procedures one at a time
    - Full Module View displays code for entire module (default) Figure 3.5 and 3.6, p. 55
  - Code:
    - 'Every time this button is clicked, the message
    - '"Welcome to Visual Basic" (comments)

A Simple Program : Printing a Line Of Text on the Form

- Print "Welcome to Visual Basic!" (command)
- End Sub
- Private Sub cmdExit_Click()
  - End 'Terminate program
- End Sub

- Object box lists form and all associated objects
- Procedure box lists procedures associated with object displayed in object box

A Simple Program : Printing a Line Of Text on the Form

- Print statement
  - prints text displayed on next line
  - default is to print on Form
- Problem?
  - Will be covered up by Control
- End statement
  - terminates program execution
  - places user in design mode
  - Note:use only 1 End for normal termination

A Simple Program : Printing a Line Of Text on the Form

- When typing a line of code
  - after entering will be checked for syntax errors
  - Figure 3.8, p. 57 syntax error dialog
- Syntax errors
  - violation of language rules
- Color scheme :(Tools>Options>Editor Format)
  - Comments-green
  - events-black
  - reserved words-blue

Another Simple Program: Adding Integers

- Program receives Integers from user, computes sum, and displays result
  - Figure 3.10, p. 60
- Object properties: Figure 3.11, p. 60-61
- Program code: Figure 3.12, p. 61
- Textbox control is main user input
  - common properties:
  - Text=stores text
  - Enabled=True (default)

Another Simple Program: Adding Integers

- Note1: if False, user cannot interact with textbox
- Note2: text representing sum appears gray
  - indicates it is disabled
- MaxLength=limits how many characters can be entered in textbox
- default=0

- General Declaration
  - statements are available to every event procedure
  - Example: Dim sum As Integer

Another Simple Program: Adding Integers

- Sum used to store information
- Note1: variable names cannot be keywords and must begin with a letter
  - will cause syntax error if does not begin with a letter
  - can be made up of letters, numbers, and underscore
- Note2: Visual Basic is not case sensitive
- Dim explicitly (formally) declares variables
- As part of declaration and variable types

Another Simple Program: Adding Integers

- Example: Integer would be any whole number
- Integers are stored in two bytes,
  - range -32767 to 32768
  - default value is 0
- Note1: choose meaningful variable names
- Note2: exceeding and Integer's range is a run-time error
- Variables can be declared using special symbols: type declaration characters

Another Simple Program: Adding Integers

- Example: Dim sum% (integer TDC)
  - Figure 5.3, p. 165
  - someVariable% = 8 (implicitly declares integer)
- Note1: variables exist only as long as procedure is executing
- Note2: TDC with As is a syntax error
- variable not given a type is given default type variant
- Dim x As Integer, Y

Another Simple Program: Adding Integers

- Dim x,y As Integer
- Access a property:
  - use object's name followed by period and property name
  - sum = sum + txtInput.Text
  - expressions or values that cannot be implicitly converted result in run-time errors
- Let sum = sum + txtInput.text
  - Let is optional

Another Simple Program: Adding Integers

- txtInput.Text = " " (clears object)
- txtSum.Text = sum (converts sum's value to string)
- Memory Concepts
  - variables correspond to storage locations in computer's memory(RAM)
  - value assigned to variable replaces current value stored (destructive read-in) Figure 3.13, p. 64
  - value read out of memory , current value is unaffected (nondestructive)

Arithmetic Operations and Operator Precedence

- Arithmetic:
  - operations summarized in Figure 3.14, p. 65
  - Mod (modulas) operator returns Integer remainder after Integer division
    - rounds any fractional part before performing operation
    - example: 20 Mod 5 = 0 ; 7 Mod 4 = 3
  - Note1: integer division is rounded before division takes place (round up)
  - Note2: Floating-point division yields a floating-point number (represents real numbers)

Arithmetic Operations and Operator Precedence

- Order Precedence:
  - ()
  - ^ exponentiation
  - - negation
  - */
  - \ integer division
  - Mod
  - +-
  - Example: z = p * r ^ q + w / x - y (Figure 3.16, p. 68)

Arithmetic Operations and Operator Precedence

- Parentheses can make a complex expression clearer
- Decision making Comparison Operators:
  - used in If/Then structure (True/False)
  - Condition True- execute statement(s)
  - Condition False- statement(s) are not executed
  - Operators: Figure 3.17, p. 69
    - d = g; d is equal to g
    - s <> r; s is not equal to r ***

Arithmetic Operations and Operator Precedence

- y > i; y is greater than I
- p < m; p is less than m
- c >= e; c is greater than or equal to c ***
- m <= s; m is less than or equal to s ***
- *** reversing order of symbols will cause syntax error

- Gui interface comparing two numbers
  - Figure 3.18, 3.19 ; p. 70-71
  - Code: Figure 3.20, p. 72
- Option Explicit- forces variables to be explicitly declared (in General Declarations)

Decision Making Comparison Operators

- Tool>Options>Editor tab
  - check Require Variable Declaration
- Full Module view: Figure 3.20, p. 55
- Tab Width: set number of spaces correspond to tab
- Function InputBox displays an input dialog
  - user is prompted to click OK or Cancel to make dialog box go away
  - InputBox(prompt, title)

Arithmetic Operations and Operator Precedence

- String concatenation operator is &
  - strings information together
- Each control has a default property
  - Example: for Label is Caption
- Note1: write each If/Then structure on multiple lines using End If to terminate condition
- Note2: statements within body of If/Then should be indented
- Note3: writing property name improves reading

Arithmetic Operations and Operator Precedence

- White space characters are ignored by the compiler
  - Example: tabs and spaces
  - Exception: except when inside double quotes
- Sentences may be split over several lines if line-continuation character, _, is used
  - if not used, results in syntax error
  - minimum of one white-space character must precede line-continuation character

Arithmetic Operations and Operator Precedence

- Placing anything after line-continuation character is a syntax error
- Use colon : to combine two statements on a single line
- square = number ^ 2 : cube = number ^ 3
- Splitting an identifier or keyword is syntax error
- writing one statement per line improves program readability