FIT 100

Functions

FIT 100

Functions and Procedures

- Similarities:
  - Little mini-programs that are named and include a series of code statements (instructions) to be executed when called.
- Differences:
  - Functions have a specific answer (value) to be returned to the caller when they finish
- Functions are often created by a programmer for computing formulas or performing common math operations.
  - This would indicate that they can have parameters and accept arguments
- What is nice for us is Visual Basic has already built in many of the most common functions. All we have to do is call them!!!!

FIT 100

Built In Functions in VB

- You have already seen 2 common functions used in Visual Basic
  - You used them in your first VB lab
- What are they?
  - Remember, a function, when called, returns a specific value to the calling location of the program
- Functions can take none, one or many parameters
  - Just like procedures

FIT 100

Built In Functions

- String
  - Return a string that is passed as a parameter all in UPPER CASE, or lower case
  - Return a particular number of string characters starting on the left side, or right side, or in between
  - Count the number of characters in a string
  - And many more…
- Conversion
  - Return a string value when given a number
  - Return a number when given a string (if possible)
  - Convert a string value to a date format

FIT 100 Built In Functions

- Mathematical
  - Returning the integer portion of a particular number
  - Returning the square root of a number
  - Rounding a number to the nearest Integer value
- Miscellaneous
  - Return the current system date
  - Return the current system time
  - Return a pseudo random number that is greater than 0 and less than 1
  - Return a specific intensity of light (color)
  - Return a string value with a particular format:
    - Currency, Date, Percent, etc.

FIT 100 Calling a Built In Function

- Built in functions will return a specific value
  - Calling a function is slightly different than calling a procedure
  - Calling incorporates assignment
- The value to be returned must be assigned to a variable or object property that will hold the same data type

```
lblALabel.Backcolor = RGB (0, 255, 0)

Dim varColor As Double
varColor = RGB (255, 255, 0)
```

A **function call** can be used anywhere Visual Basic expects a value whose type is the same as the **function return value** type.

- Function calls like the ones above MUST go on the right side of assignments. NEVER on the left

FIT 100 More Built-In Function Calls

```
Dim x as Double
Dim y as Integer
Dim z as String
....
```

x = RGB (0, 0, 255) frmTemp.BackColor = x	frmClock.Caption = Date
y = 150 x = RGB (0, y, 255)	lblALabel.Caption = Time
y = Int (Rnd( )*y) y = Int (x)	z = “HELLO WORLD!” z = LCase (z)

What Have You Learned About Programming So Far?

FIT 100

Let’s review:

- Variables
- Expressions
- Conditionals
- Procedures

FIT 100 Variables

- Variables
  - Locations in memory
- Variable names
  - The way we refer to the locations in memory in our program
- Variable declaration
  - Listing the names of variables to be used in program
- Data types of variables
  - String, Integer or Double - there are other types but we won't cover them in this course
- Variable initialization
  - Assigning a value to a variable to begin with so that we control content
- Variable values
  - The data stored in those memory locations, subject to change
- Assignment statements
  - The command to change the value of a variable

<Variable Name> <Assignment Symbol> <Expression>

  - x= 127 or x = x + 1

FIT 100 Expressions

- A means of performing the actual computation
- Many kinds of expressions. They can include:
  - logical operators: And, Or, Not
  - relational operators: <, >, <=, >=, < >
    - When used here = means test to see of operands are the same
  - binary operators: +, *, &
  - unary operators : -, ^, Not

FIT 100 Conditionals

- Used when a decision must be made between one or more possibilities (conditions)
- Basic conditional
  - If <T/F Statement> Then ‘tests for one condition: true
- General conditional
  - If <T/F Statement> Then ‘tests for one condition, allows 2 outcomes. One for True,
    <code statements>
    Else the other for False (or otherwise)
    <code statements>
    End If
  - If <T/F Statement> Then ‘tests for multiple conditions
    <code statements>
    Elseif <T/F Statement> Then
    <code statements>
    …
    Else
    <code statements>
    End If

FIT 100 Conditionals

```
gradePt = 4.O
If passClass = true then
    If theLetterGrade = “A” then
        lblGrade.Caption = “You got a “ & gradePt
    Else
    lblGrade.Caption = “You didn’t quite get a “ & gradePt & “, but you passed!”
    End If
Else
    lblGrade.Caption = “You did not pass and are nowhere near a ” & gradePt
End If
```

- Take out a piece of paper
- What does this program put into lblGrade.Caption if the variables have the following values:
  - A) passClass = false; theLetterGrade = “A”;
  - B) passClass = true; theLetterGrade = “C”
  - C) passClass = true; the LetterGrade = “A”

FIT 100 Adding Another Condition: ElseIf

- The conditional statement (If-Then-Else) is one way you know, so far, to control which statements are executed.
- In VB6, using ElseIf is a way to test a long sequence of possible conditions:

```
If <T/F condition> Then
      <code statement list>          'code statements for 1st condition
ElseIf <T/F condition> Then
      <code statement list>          'code statements for 2nd condition
ElseIf <T/F condition> Then
      <code statement list>          'code statements for 3rd condition
....
Else
      <code statement list>          'code statements for "otherwise"
End If
```


FIT 100 Potential Problems with ElseIf

- An If statement that uses ElseIf passes through all of the previous cases before reaching a given test. What are the consequences of this?

```
If num > 10 Then
  result = "More than 10"
Elseif num > 20 Then
  result = "More than 20"
Else
  result = "Less than or equal to 10"
End If
```

- Will the ElseIf statement ever be executed?

FIT 100 Mini-Exercise #1

- What is the value of x after the form has been loaded?

```
Option Explicit
Dim x As Integer

Private Sub squid()
   x=x+2
End Sub

Private Sub Form_Load
   x=0
   Call squid
End Sub
```


FIT 100 Mini-Exercise #2

- What is the value of y after the form has been loaded?

```
Option Explicit
Dim y As Integer

Private Sub squid()
   y=y+2
End Sub

Private Sub clam()
   call squid
   call squid
End Sub

Private Sub Form_Load
   y=0
   Call squid
   Call clam
End Sub
```

FIT 100 Remember Procedure Structure

- Parts of a procedure specification
  - Name
  - Definition
  - Parameters
  - Declaration

```
Private Sub calcRecArea (base as Integer, height as Integer, _
                         area as Integer)
        area = base * height
End Sub
```


FIT 100 Input vs. Output

- Many programming languages (including VB6) provided several different ways of passing values back and forth between the actual and formal parameters
- The default in Visual Basic is **pass by reference**
- Pass by reference allows information to flow in both directions.
  - Formal parameters can be used as inputs or outputs or both
  - Any changes made to a formal parameters will make a change to the corresponding actual parameter
  - Remember the Temp Conversion program from last class?

FIT 100 Actual Parameters

- The actual parameters must follow these formal/actual correspondence rules
  - Same number of actual parameters as formal parameters in the procedure declaration
  - Order matters!
    - The 1st actual parameter corresponds to the 1st formal parameter
    - The 2nd actual parameter corresponds to the 2nd formal parameter
    - Etc, etc, etc
  - Data types of actual parameters must match data types of formal parameters
  - Any formal parameter used as a procedure output must have a variable for the corresponding actual parameter

FIT 100 Mini-Exercise #3

- What is the value of y after the form has been loaded?

```
Option Explicit

Private Sub Form_Load()
   Dim y As Integer
   y=0
   Call squid(1, y)
   Call clam (2, y)
End Sub

Private Sub clam(dork As Integer, zebra As Integer)
   call squid (dork, zebra)
    dork = zebra + 2
   call squid(dork, zebra)
End Sub

Private Sub squid(x as Integer, z As Integer)
   z = x+2
End Sub
```

FIT 100 Exercise # 4

- Given the following procedure declaration:

```
Private Sub example(r As Double, area As Double)
      area = 3.1415926 * r ^ 2
End Sub
```

and the following statements elsewhere in the program:

```
…
value1=10
value2= 5
Call example(value1, value2)
…
```

Write a statement with the same affect as the Call statement

FIT 100 Hmmmm, How Is It Done?

- For Wednesday, think about writing a program to do the following:

10 seconds
9 seconds
8 seconds
7 seconds
6 seconds
5 seconds
4 seconds
3 seconds
2 seconds
1 seconds
Blast Off!!!!!