Basics of Procedures

FIT 100

Procedures encapsulate useful computation in a form that can be reused. In this regard they extend the capability of the computer since the procedure can be used as if it were a primitive instruction.

FIT 100 Importance of Procedures

- Procedures encapsulate functionality so that it can be reused. This will be the primary emphasis in today's lecture.
- Another important aspect: procedures help manage complexity. This aspect becomes obvious only when you start writing much more complex programs.

FIT 100 A Scenario: Reading Email

- You are reading email and your friend living outside the US says the temperature is 38º
- That's Celsius, of course. What is it in Fahrenheit? Is it hot or cold, you wonder. Why doesn't your computer have a Celsius-to-Fahrenheit converter?
- This situation arises all of the time … there are many things a computer could do for you, but the software is not available
  - You can step through the process yourself, i.e. convert to Centigrade
  - But what you'd like is to solve the problem once-and-for-all and have the solution packaged-up to be always available
- What you want is a procedure

FIT 100 The Idea of Procedures

- Procedures encapsulate computation for general application
  - *A procedure's operation should be hidden from view*
  - It must be possible to give data to a procedure and get results back from the procedure
  - All of the possible eventualities must be considered
- The procedure concept has two parts:
  - A procedure "declaration" -- defines how computation goes
  - Many procedure "calls" -- requests to have the procedure performed

The fundamental idea of procedures: Whenever the procedure is called, "substitute" its definition

FIT 100 Anatomy Of A Procedure

- Procedures have the following features
  - Name, a brief description of operation performed
  - Parameters, variables used for passing input in, output out
  - Body, the statements that perform the desired computation
- The VB6 procedure to convert Celsius to Fahrenheit
  - Name is c2f (Snyder book uses convertC2F … shortened to fit on slides …)
  - Parameters: input is c; output is f
  - Body is standard conversion equation
  - Blue -- key words and and symbols that are required

```
Private Sub c2f (c As Integer, f As Double)
        f = 9 * c / 5 + 32
End Sub
```


FIT 100 Using the c2f Procedure

```
…
` Sydney temp is 38
Call c2f(38, s)


Msgbox("Temp is "
        & s)
```

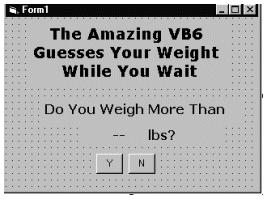
At the start of the procedure:
c gets 38

At the end of the procedure:
s gets 100.4

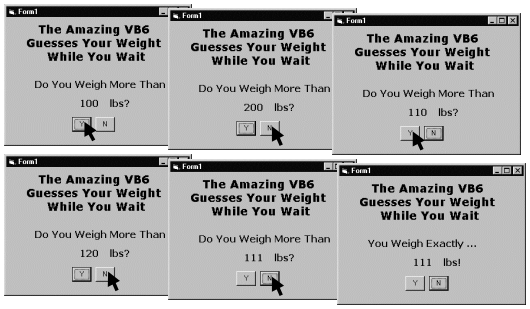
```
Private Sub c2f (c As Integer,
                 f As Double)
      f = 9 * c / 5 + 32
End Sub
```

FIT 100 A Guessing Game

- Develop a program to guess a person's weight
  - It starts with a guess of 0 and always stays below the correct answer
  - A weight guess is formulated as: loSide + increment
  - Questions are asked in increments of 100, then 10, then 1



FIT 100 Operation ...



FIT 100 Braining Out The Logic

- When will guesses be made?
  - Initially, when the program begins (called ***form_load***)
  - In response to a Yes answer
  - In response to a No answer
- In addition to the first guess what happens at start
  - Initialize `loSide = 0`
    `increment = 100`
- In addition to a guess, what happens on a Yes?
  - Add-in increment, as weight is more than `loSide + inc`
- In addition to a guess, what happens on a No?
  - Reduce the increment by dividing by 10
  - Check if the increment is below 1 … that'll be the answer

FIT 100 Including A Procedure

- The fact that a guess must be made in three places is motivation to define a procedure to make the guess (despite the fact that it is a trivial computation)

```
Option Explicit
Dim loSide As Integer
Dim increment As Integer
-------------------------------------------
Private Sub guess()
    lblGuess.Caption = loSide + increment
End Sub
-------------------------------------------
Private Sub Form_Load()
    increment = 100
    loSide = 0
    Call guess
End Sub
```

`lblGuess.Caption = loSide + increment`

FIT 100 The Yes/No Logic

- The "Yes" logic only adds-in, but the "No" logic reduces the increment and must also test for completion

```
Private Sub cmdYes_Click()
    loSide = loSide + increment
    Call guess
End Sub
-------------------------------------------
Private Sub cmdNo_Click()
    increment = increment \ 10
    If increment < 1 Then
       lblHead.Caption = "You Weigh Exactly ..."
       lblGuess.Caption = loSide
       lblPound.Caption = "lbs!"
    Else
        Call guess
    End If
End Sub
```

`lblGuess.Caption = loSide + increment`

FIT 100 Procedural Abstraction

- Whenever the same operations are performed in different places in a program, there is an opportunity for *procedural abstraction*
- Procedural abstraction gives a name to the operations
- It also encapsulates the operations so they can be executed out-of-view, receiving input via parameters and influencing the calling environment only by the result(s) returned

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 #1 -- Answer

- 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
```

X = 2

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 Mini-Exercise #2 -- Answer

- 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
```

y = 6