Procedural Basics

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.

A Scenario: Reading Email

- You are reading email and your friend living outside the US says the temperature is 38°
- 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 C
    - 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

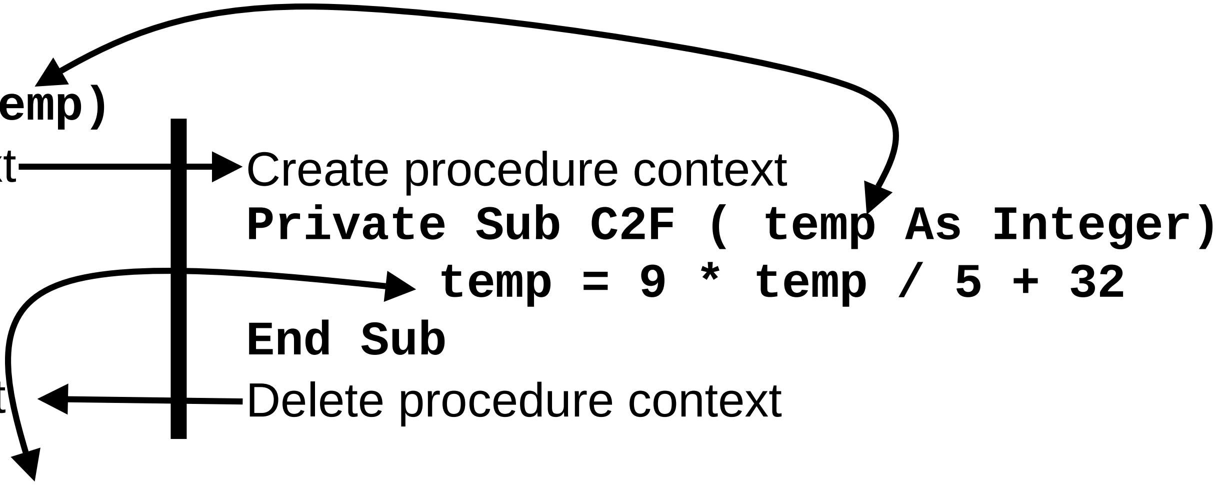
Anatomy Of A Procedure

- 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
  - Parameter -- both input and output -- temp
  - Body is standard conversion equation
  - Blue -- key words and and symbols that are required

```
Private Sub C2F ( temp As Integer)
            temp = 9 * temp / 5 + 32
End Sub
```

Tale Of Two Contexts

- ❖ There is a calling context that is suspended when a procedure is called and the procedure context that comes into existence on the call, and vanishes on completion when the calling context is resumed



```
…
Sydney_temp = 38
Call C2F(Sydney_temp)
```

Suspend calling context

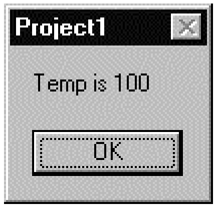
Create procedure context

```
Private Sub C2F ( temp As Integer)
      temp = 9 * temp / 5 + 32
End Sub
```

Delete procedure context

Resume calling context

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



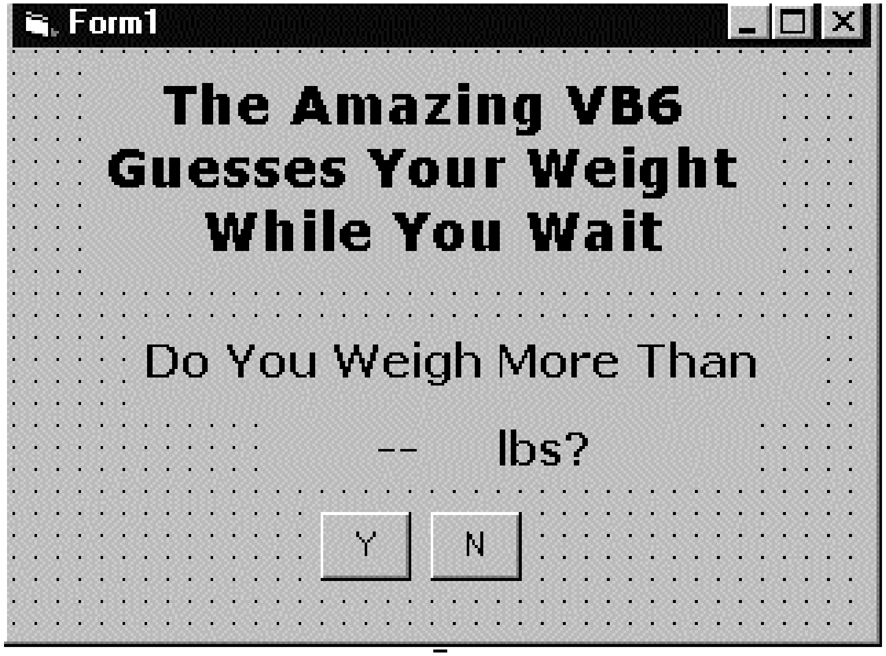
Temp is 100

```
...
```

⇐*Calling Context Procedure Context*⇒

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

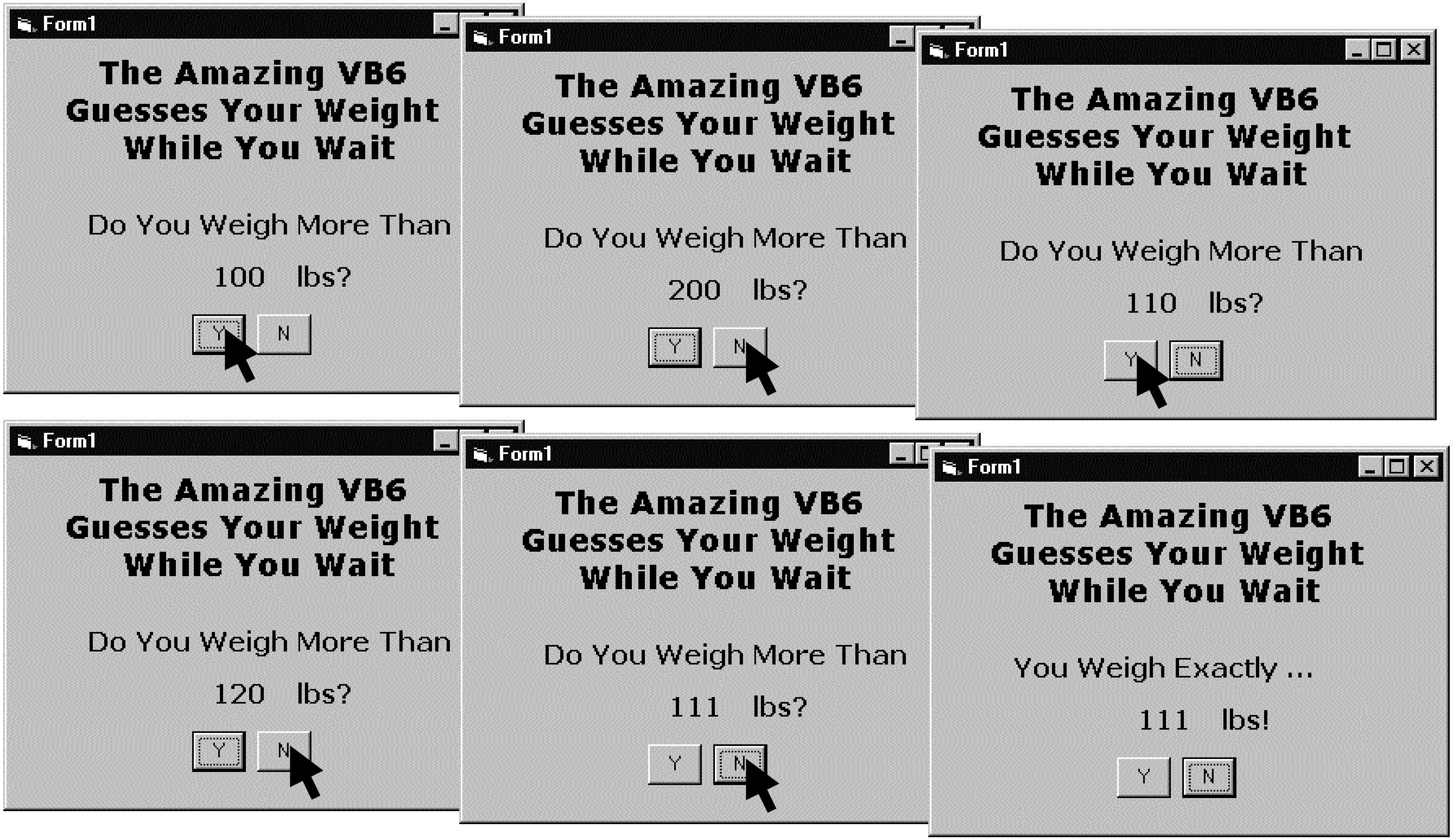


The Amazing VB6 Guesses Your Weight While You Wait

Do You Weigh More Than

-- lbs?

Operation ...



The Amazing VB6
Guesses Your Weight
While You Wait

Do You Weigh More Than

100 lbs?

The Amazing VB6
Guesses Your Weight
While You Wait

Do You Weigh More Than

200 lbs?

The Amazing VB6
Guesses Your Weight
While You Wait

Do You Weigh More Than

110 lbs?

The Amazing VB6
Guesses Your Weight
While You Wait

Do You Weigh More Than

120 lbs?

The Amazing VB6
Guesses Your Weight
While You Wait

Do You Weigh More Than

111 lbs?

The Amazing VB6
Guesses Your Weight
While You Wait

You Weigh Exactly ...

111 lbs!

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

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
```

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
```

```
lblGuess.Caption = loSide + increment
```

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

```
lblGuess.Caption = loSide + increment
```

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