Procedures

Procedures are a part of our everyday lives. Individuals and organizations utilize them as a way to assure that a task is performed in a thorough and predictable manner each time it is needed.

Computers also use procedures in this manner. Procedures encode the operations needed to accomplish a task. In other words, procedures encode algorithms.

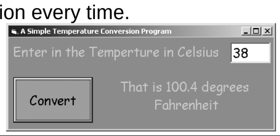
Importance of Procedures

- ❖ Procedures encapsulate functionality (useful instruction) so that it can be used anywhere, anytime.
- ❖ Procedures help manage complexity
  - ☐ If you find yourself writing the same code statements multiple times in your program, this is a good indication that you need a procedure to minimize the amount of code.

A Simple Scenario

- ❖ We use email every day to send mail to friends in the state, across the country or around the world.
- ❖ You receive mail from your friend in Australia telling you it's 38º
- ❖ The temperature is Celsius, but you want Fahrenheit
- ❖ You could do a quick calculation, but since you write to this person a lot, it would be better to just write a little procedure to do the calculation every time.

Application vs. Procedure: The application is the entire facility (GUI + computation). The procedure is just the computation


A Simple Temperature Conversion Program

Enter in the Temperture in Celsius 38

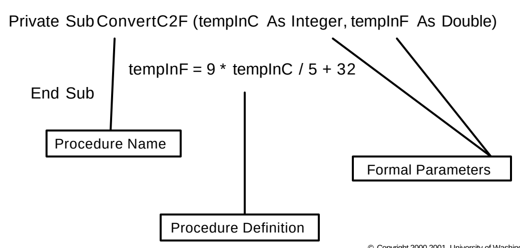
That is 100.4 degrees Fahrenheit

Structure Of A Procedure

- ❖ Procedures encapsulate (package up) a computation to be used anywhere, anytime
- ❖ Procedures have the following features:
  - ☐ Name: term used to refer to the task the procedure performs,
    example: convertC2F
  - ☐ Definition: The steps that will accomplish the task. Also known as the *procedure body*
    tempInF = 9*tempInC / 5 + 32
  - ☐ Parameters: the names of the input data and the output results
    tempInC As Integer, tempInF As Integer
  - ☐ Declaration: the entire package of the name, definition and parameters

The Whole Procedure Package

- Procedure Declaration: Includes name, definition and parameters



```
Private Sub ConvertC2F (tempInC  As Integer, tempInF  As Double)
        tempInF = 9 * tempInC / 5 + 32
End Sub
```

You Have Already Seen Procedures!

- The event handling routines that you have manipulated with code (i.e. the click events) are procedures
- The Private Sub part of the procedure is a reference to the old name for procedures within a program: Sub Routines, shortened to Sub in the program language notation
  - Private Sub cmdOK_Click()

    End Sub
- Every time a user activates a control, i.e. clicks a command button, VB calls the procedure needed: the event handling routine

Calling A Procedure

- The procedure declaration only specifies how the procedure works and only needs to be given once
- The procedure call says when, where and with what values the procedure will be performed (executed)
  - A procedure call can be used anywhere that the task to be performed is needed
- Call convertC2F (38, degreesF) is a VB procedure call specifying the *procedure to be executed* (convertC2F) and the *values to be used* (38 is the Celsius temperature input and degreesF is the variable for the result)

```
Private Sub ConvertC2F (tempInC As Integer, tempInF As Double)
        tempInF = 9 * tempInC / 5 + 32
End Sub
```

Parameter Correspondence

- The parameters name the input values and the output results to the procedure
- The number of parameters in the declaration must match the number of parameters in the call, and they correspond one-to-one
- The parameters are referred to by separate names
  - Formal parameters are parameters of the declaration
  - Actual parameters are parameters of the call

```
Call convertC2F (38, degreesF)

Private Sub ConvertC2F (tempInC As Integer, tempInF As Double)
        tempInF = 9 * tempInC / 5 + 32
End Sub
```

FIT 100 What Happens…

- A procedure call “makes it happen”
- Substitution Rule: The procedure call operates as if the definition replaces the call and the actual parameters replace the formal parameters

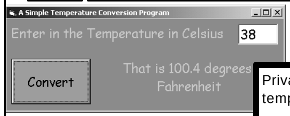
```
Private Sub ConvertC2F (tempInC As Integer, tempInF As Double)
          tempInF = 9 * tempInC / 5 + 32
End Sub
```

… Code of a VB Program

~~Call convertC2F (38, degreesF)~~ degreesF = 9*38 / 5 +32

…

FIT 100 Calling the convertC2F Procedure


A Simple Temperature Conversion Program
Enter in the Temperature in Celsius
38
Convert
That is 100.4 degrees Fahrenheit

```
Private Sub ConvertC2F (tempInC As Integer,
tempInF As Double)
   tempInF = 9 * tempInC / 5 + 32
End Sub
```

```
Private Sub cmdConvert_Click()
  Dim degreesC As Integer
  Dim degreesF As Double
  degreesC = txtTempC.Text

        Call ConvertC2F(degreesC, degreesF)

   lblTempF.Caption = "That is " & degreesF & " degrees Fahrenheit"
End Sub
```

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 returning results or creating effects at the point of the call

FIT 100 Summary

- Procedure declarations encapsulate name, parameters and definition
- Procedure calls cause the procedure to be executed
- Parameters must match in number and order
- The Substitution Rule defines how procedures work