General Procedures and Functions

Procedures

A procedure is a separate piece of the program that is named and can be executed by calling it from other parts of the program. The procedure can be given values for it to work with. These values are given names in the procedure code, sometimes referred to as formal parameter names.

A **procedure definition** has the form:

<Private or Public> Sub ProcedureName (<parameter1>, <parameter2>, …)
    <some actions>
End Sub

Before the procedure name, the Public or Private keyword says whether the name of the procedure is public to other forms or private to stay within the current form.

Each <parameter> consists of either the keyword ByVal or ByRef, the name of the parameter, and the declaration of the parameter type. These are called the **formal parameters**.

Parameter types can be data types, such as Integer, Single and String, or they can be any of the form object types, such as TextBox or Label.

Example: [Note that the procedure header (the first line) doesn't fit on one line in the Word document, but it must go on one line in Visual Studio!]

```
Private Sub ComputePrice
    ( ByVal number As Integer, ByVal unitprice as Single, ByRef totalprice As Single)
      totalprice = number * unitprice
End Sub
```

The ByVal keyword is used when values will be used in the procedure, and the ByRef keyword is used for parameters that will be assigned to.

A **procedure call** has the form: Call ProcedureName(paramvalue1, paramvalue1, …). The values paramvalue1, paramvalue2, etc. are called the **actual parameters**.

- The number and types of the actual parameters must match the number and types of the formal parameters in the procedure definition.
- The actual parameter for a ByRef parameter must either be a variable or an Object, such as a Label or TextBox.

Example:

```
    Dim total as Single
```

```
Call ComputePrice ( 15, 2.00, total)
 Label1.Text = CStr(total)
```

As this the program executes the procedure call, it jumps to the ComputePrice code and gives the values in the procedure call to the parameter variables names in the procedure definition. The procedure code is executed and then the flow of control jumps back to where it was called. At this point, any ByRef parameters that were assigned will have the values given to them by the procedure.

The same procedure can be called any number of times with different values for the parameters.

Functions

Functions are very similar to procedures except that the function call will also return a value to the place where the function was called.

The **function definition** uses the keyword Function instead of Sub, but the formal parameter list is the same:

```
<Private or Public> Function FunctionName ( <parameter1>, <parameter2>, …)
        <some actions> (including an assignment to FunctionName)
End Function
```

Example:

```
Private Function ComputePriceFn (ByVal number As Integer, ByVal unitprice as Single)
        ComputePriceFn = number * unitprice
End Sub
```

The crucial difference is that you must assign to the function name during the actions, and this is the value that is returned to the point of call.

The **function call** has the form: FunctionName (paramvalue1, …), which is used wherever a value is needed in the program.

Example1:

```
        Dim total as Single
        total = ComputePriceFn ( 15, 2.00)
        Label1.Text = CStr(total)
```

Example2:

```
         Label1.Text = CStr(ComputePriceFn ( 15, 2.00))
```

In this example, the call to ComputePriceFn returns a value which is converted by the CStr function and assigned to Label1.Text.