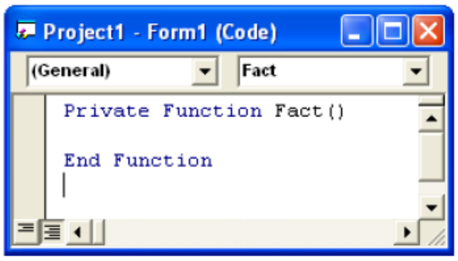
Function: function procedures and sub procedures share the same characteristics, with one important difference- function procedures return a value (e.g., give a value back) to the caller, whereas sub procedures do not. Function Procedures can be created with the add procedure dialog shown in figure by selecting function. Figure bellow shows a function procedure. **Fact**, created with the add procedure dialog. Fact implicitly returns variant.



```
Private Function Fact()

End Function
```

Fact could also have been created by typing the function procedure directly into the code window.

The line

Private Function Fact() is the function procedure header. The header contains the keyword function, the function name and parentheses. The declarations and statements that the programmer will insert between the header and **End Function** form the function procedure body, Fact is invoked with the line.

Result= Fact( ) When a function procedure name (such as **Fact**) is encountered at run time, the function procedure is called, causing its body statements to execute. Consider the complete definition for **Fact**

Private Function Fact(N)
Fact=N^2
End Function
A function procedure return value is specified in the body by assigning a value to the function procedure name, as in Fact=N^2 Then returns (along with the value returned) to the calling statement
Result=Fact (N) And the return value is assigned to variable result. Program execution then continues with the next statement after the call to Fact. All function procedure definitions contain parentheses, the parentheses may be empty (e.g. Fact) or may contain one parameter variable declarations. Consider the following function procedure:

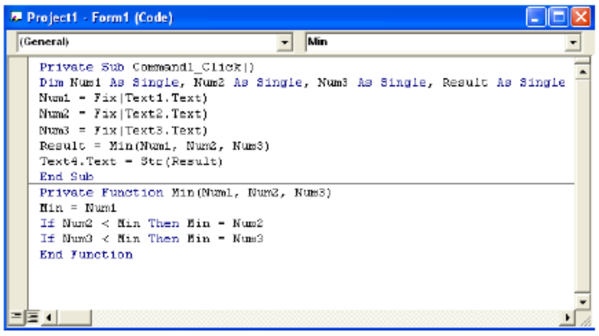
Private Function Area (s1 as single,s2 as single)
Area=s1*s2
End Function

Which declare two parameter variables s1, and s2. Area's return type is variant. Area is called with the statement Square=area(8.5, 7.34) The value 8.5 is stored in s1 and the value 7.34 is stored in s2.

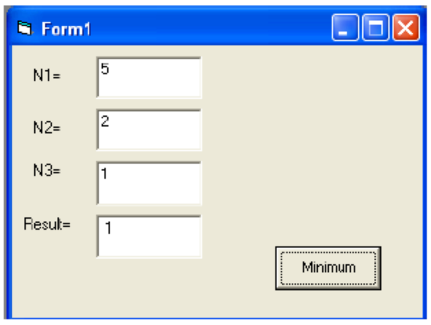
Example: Write a code program to read three integer numbers. Using a define sub Function (Min) to determine the smallest of three integers. Display the smallest value in textbox.

Solution:

```
Private Sub Command1_Click()
Dim Num1 As Single, Num2 As Single, Num3 As Single, Result As Single
 Num1 = Fix(Text1.Text)
Num2 = Fix(Text2.Text)
Num3 = Fix(Text3.Text)
Result =MinNum1, Num2, Num3)
Text4.Text = Str(Result)
 End Sub
 Private Function Min(Num1, Num2, Num3)
 Min = Num1
 If Num2 < Min Then Min = Num2
 If Num3 < Min Then  Min = Num3
End Function
```



```
Private Sub Command1_Click()
Dim Num1 As Single, Num2 As Single, Num3 As Single, Result As Single
Num1 = Fix(Text1.Text)
Num2 = Fix(Text2.Text)
Num3 = Fix(Text3.Text)
Result = Min(Num1, Num2, Num3)
Text4.Text = Str(Result)
End Sub
Private Function Min(Num1, Num2, Num3)
Min = Num1
If Num2 < Min Then Min = Num2
If Num3 < Min Then Min = Num3
End Function
```



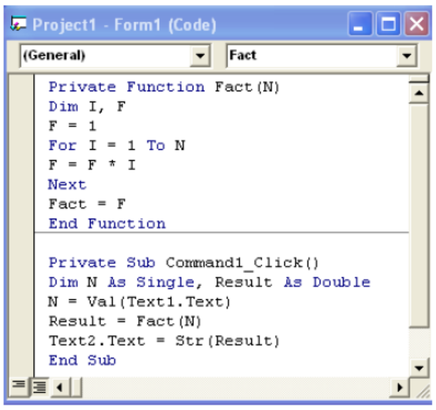
N1=	5
N2=	2
N3=	1
Result=	1

Example: Write a code program to input the value of N. Using a define sub function fact to determine(N!). Display the result into text box.

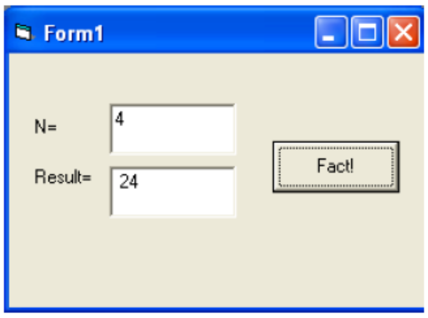
Solution:

```
Private Sub Command1_Click()
 Dim N As Single, Result As Double
 N = Val(Text1.Text)
Result = Fact(N)
Text2.Text = Str(Result)
End Sub
Private Function Fact(N)
 Dim I, F
 F = 1
```

```
For I = 1 To N
 F = F * I
Next
Fact = F
End Function
```



```
Private Function Fact(N)
Dim I, F
F = 1
For I = 1 To N
F = F * I
Next
Fact = F
End Function

Private Sub Command1_Click()
Dim N As Single, Result As Double
N = Val(Text1.Text)
Result = Fact(N)
Text2.Text = Str(Result)
End Sub
```



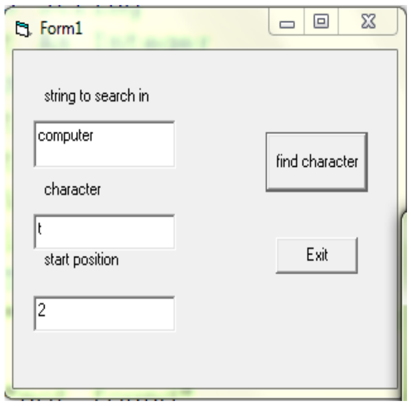
N= 4
Result= 24

<u>Example:</u> write function to find a character in the string begin the search from a determined position.

<u>Solution:</u>

```
Dim i As Integer
Function findchr(stringvalue As String, po As Integer, char As String) As Boolean
For i = po To Len(stringvalue)
  If Mid(stringvalue, i, 1) = char Then
   findchr = True
    Exit Function
  End If
 Next i
 findchr = False
End Function
Private Sub Command1_Click()
Dim bool As Boolean
Dim st As String
Dim start As Integer
Dim ch As String
st = Text1.Text
ch = Text2.Text78u
start = Val(Text3.Text)
bool = findchr(st, start, ch)
If bool Then
 MsgBox "The character is found and his position is" & i
```

```
Else
 MsgBox "not found"
End If
End Sub

Private Sub Command2_Click()
End
End Sub
```



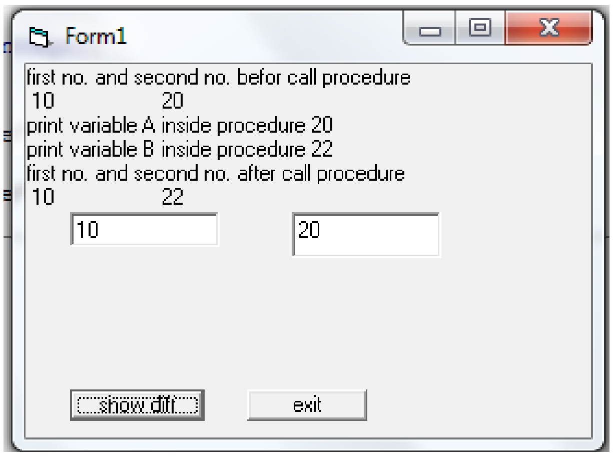
ByRef and ByVal

Parameters can be sent to a subroutine By Reference (ByRef) or By Value (ByVal). ByRef is the default, and means that changes to the variable in the subroutine will result in changes to the source variable outside of the subroutine. ByVal literally copies the values of the variables from the calling subroutine into the called subroutine. By doing this, the variables can be changed, but their values will not change outside of the called subroutine. ByVal can also be a lot slower with large variable types, however, since memory has to be copied from one location to another. If you don't have any reason to do so, there is no need to pass variables ByVal. You can explicitly state the way that a variable is passed to a subroutine by using these keywords before the variable name

Example:

```
Sub showdiff(ByVal a As Integer, ByRef b As Integer)

a = a + 10
Form1.Print "print variable A inside procedure"; a
b = b + 2
Form1.Print "print variable B inside procedure"; b
End Sub



Private Sub Command1_Click()
Dim no1 As Integer
Dim no2 As Integer
```

```
no1 = Val(Text1.Text)
no2 = Val(Text2.Text)
Print "first no. and second no. befor call procedure"
Print no1, no2
Call showdiff(no1, no2)
Print "first no. and second no. after call procedure"
Print no1, no2
End Sub

Private Sub Command2_Click()
End
End Sub
```



```
first no. and second no. befor call procedure
 10            20
print variable A inside procedure 20
print variable B inside procedure 22
first no. and second no. after call procedure
 10            22
```

10
20