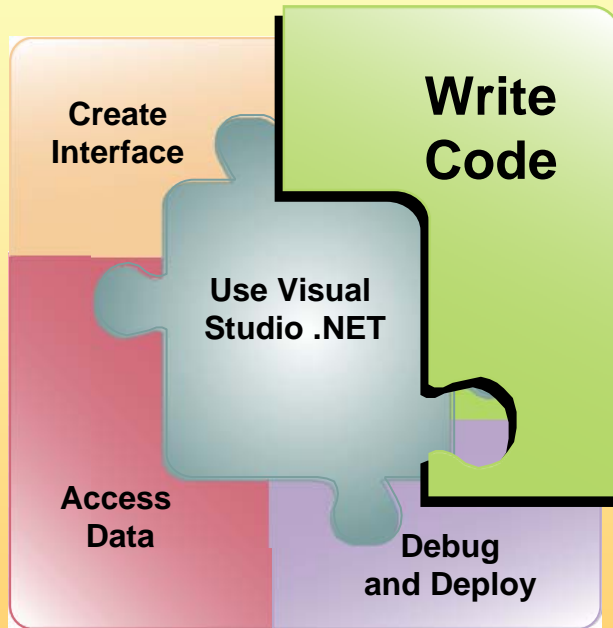


**Module 4:
Working with
Procedures**

Overview



- Creating Procedures
- Using Procedures
- Using Predefined Functions

Lesson: Creating Procedures

- What Are Procedures?
- How to Create Sub Procedures
- How to Create Function Procedures
- How to Declare Arguments in Procedures
- How to Use Optional Arguments
- Code Reusability

What Are Procedures?

- Procedures are the executable code statements in a program, enclosed by a declaration statement and an End statement
- Three types:
 - Sub procedures (including event Sub procedures)
 - Function procedures
 - Property procedures
- Enable code reuse
- Declared as public by default

How to Create Sub Procedures

Sub procedures perform actions but do not return a value to the calling procedure

```
[accessibility] Sub subname(argumentlist)  
    ' Sub procedure statements  
End Sub
```

Example:

```
Private Sub AboutHelp( )  
    MessageBox.Show("MyProgram V1.0", "MyProgram Help")  
End Sub
```

How to Create Function Procedures

Function procedures perform actions and can return a value to the calling program

```
[accessibility] Function name(argumentlist) As datatype  
    ' Function statements, including optional Return  
    ' statement  
End Function
```

Example:

```
Public Function DoubleTheValue(ByVal J As Double) As _  
    Double  
    . . .  
    Return J*2  
    . . .  
End Function
```

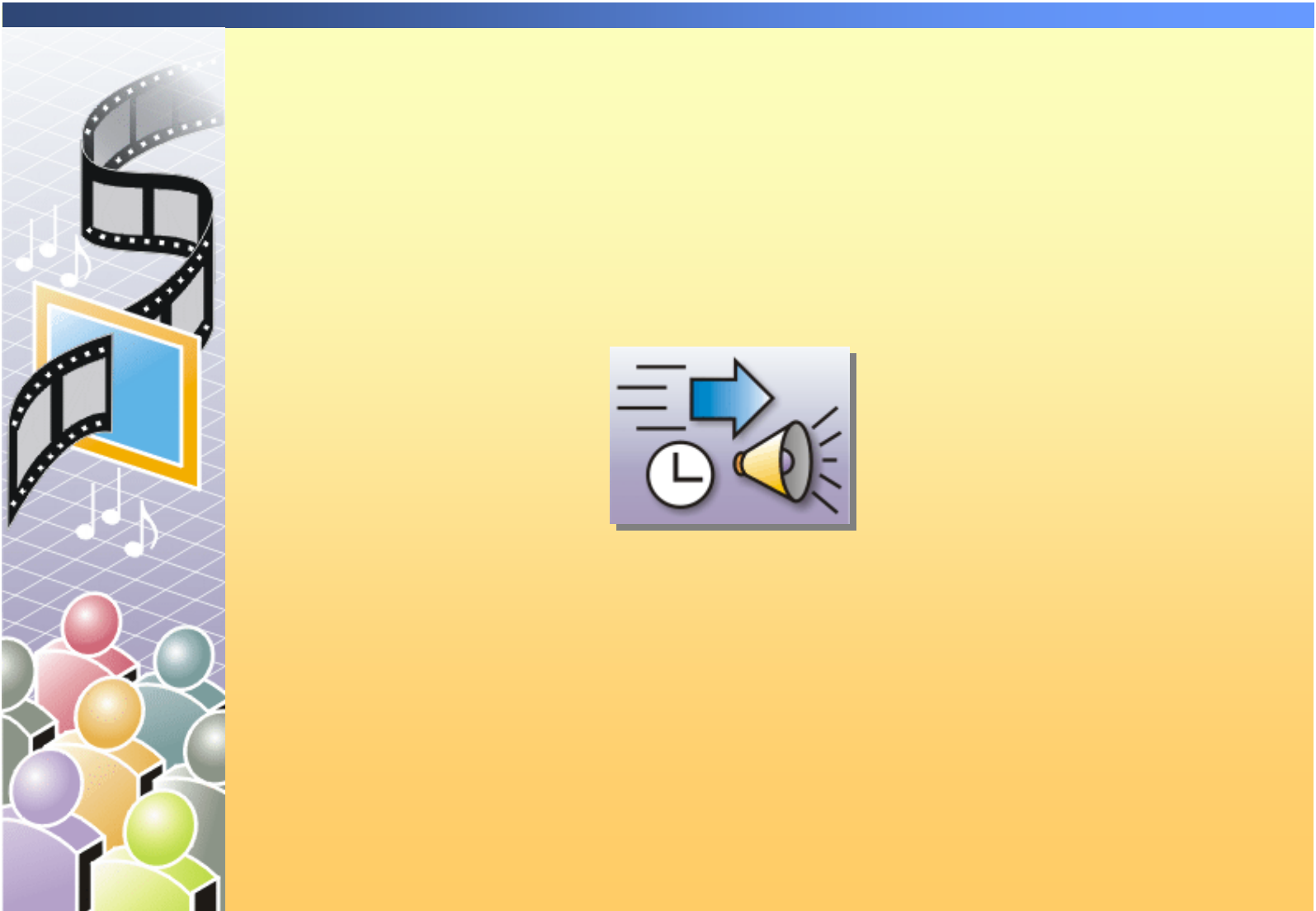
How to Declare Arguments in Procedures

- *Arguments* are data passed to procedures
- You can pass arguments *ByVal* or *ByRef*
 - **ByVal**: The procedure cannot modify the value of the original variable
 - **ByRef**: The procedure can modify the value of the original variable
 - **Exception**: Nonvariable elements are never modified in calling code, even if passed by reference
- **ByVal** is the default in Visual Basic .NET
- Syntax and example:

```
([ByVal|ByRef] argumentname As datatype)
```

```
(ByVal Name As String)
```

Multimedia: Passing Arguments



How to Use Optional Arguments

■ Rules for declaring optional arguments:

- You must specify a default value
- The default value must be a constant expression
- Arguments following an optional argument must also be optional

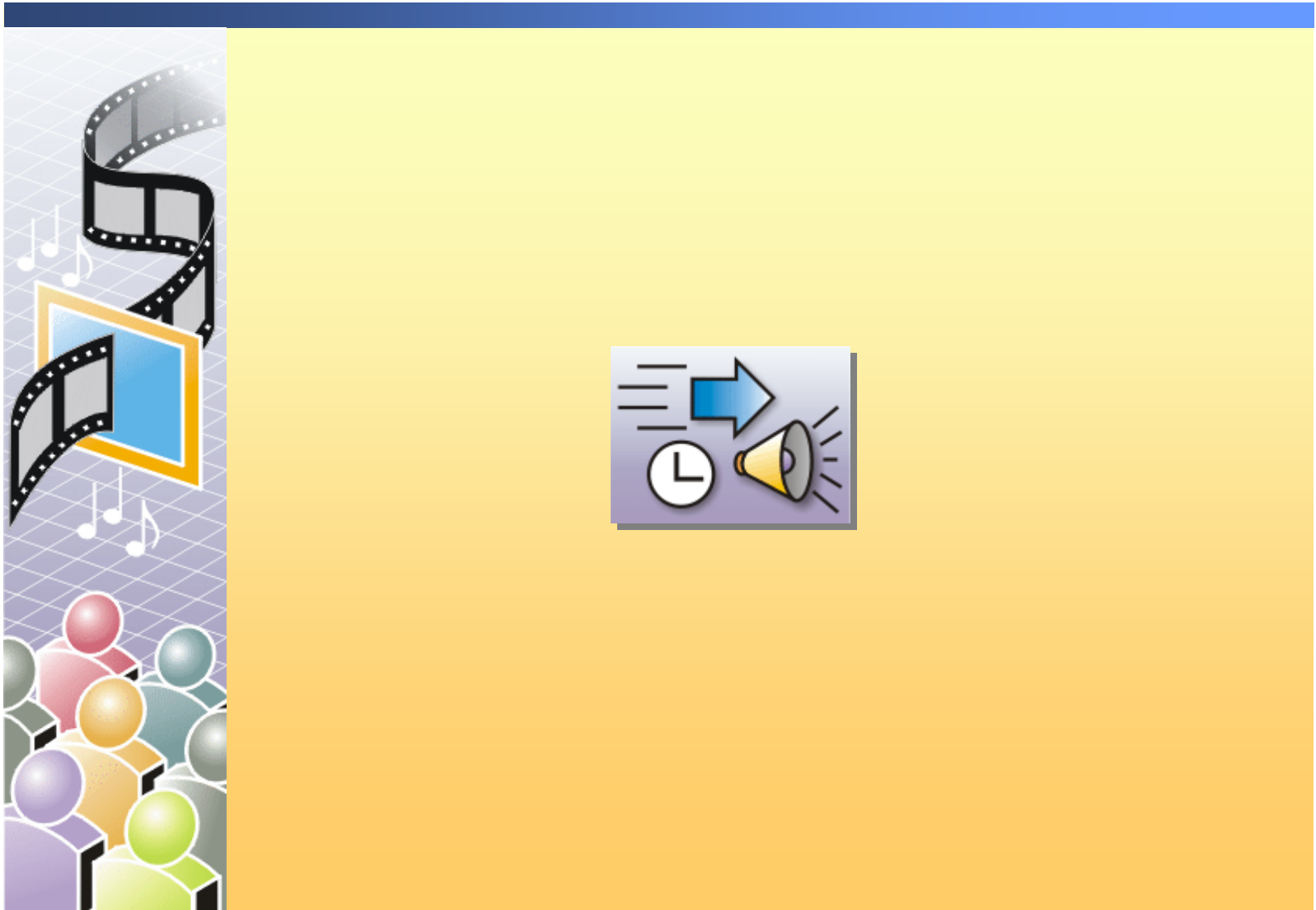
■ Syntax:

```
(Optional [ByVal|ByRef] argumentname As datatype = defaultvalue)
```

■ Example:

```
Function Add (ByVal value1 As Integer, ByVal value2 As _  
Integer, Optional ByVal value3 As Integer = 0) As Integer
```

Multimedia: Writing Reusable Code



Code Reusability

Use a...	For...	Examples
Structure	Objects that do not need to be extended	Size Point
Module	Utility functions and global data	Temperature conversion
Class	Extending objects or objects that need cleanup	Forms Button

- Creating a module:

```
[Public|Friend] Module ModuleName  
    .  
    .  
    .  
End Module
```

Practice: Creating a Function in a Module



1 Open a project

2 Add a module to the project

3 Create a function in the module

4 Write the code for the function

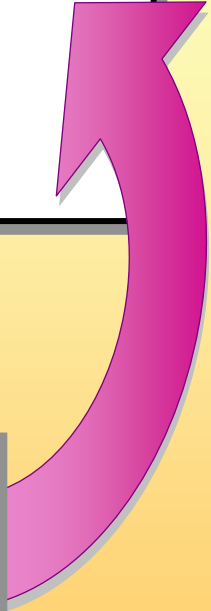
Lesson: Using Procedures

- How to Use Sub Procedures
- How to Use Function Procedures
- How to Pass Arrays to Procedures
- How to Create a Sub Main

How to Use Sub Procedures

```
Public Sub Hello(ByVal name As String)
    MessageBox.Show("Hello " & name)
End Sub
```

```
Sub Test( )
    Hello("John")
End Sub
```



How to Use Function Procedures

- Calling a function

- Include the function name and arguments on the right side of an assignment statement

```
Dim celsiusTemperature As Single  
celsiusTemperature = FtoC(80)
```

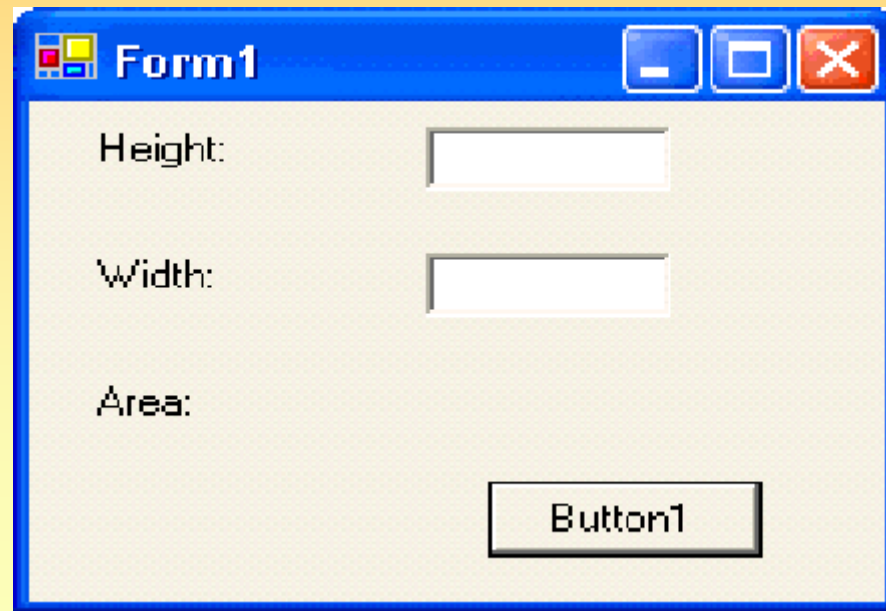
- Use the function name in an expression

```
If FtoC(userValue) < 0 Then  
    ...  
End If
```

Practice: Using the Return Value of a Function

1 Create the user interface

2 Write code for the application



The image shows a screenshot of a Windows application window titled "Form1". The window has a blue title bar with standard minimize, maximize, and close buttons. The main content area is light gray and contains three labels with corresponding input fields: "Height:" followed by a text box, "Width:" followed by a text box, and "Area:" followed by a text box. At the bottom right of the form, there is a button labeled "Button1".



How to Pass Arrays to Procedures

- Pass an array as you pass other arguments:

```
Sub PassArray(ByVal testScores As Integer( ))  
    ...  
End Sub  
  
Dim scores( ) As Integer = {80, 92, 73}  
PassArray(scores)
```

- Declare a parameter array:

```
Sub StudentScores(ByVal name As String, ByVal _  
    ParamArray scores( ) As String)  
    ' Statements for Sub procedure  
End Sub
```

- Call a procedure with a parameter array:

```
StudentScores("Anne", "10", "26", "32", "15", "22", "16")
```

How to Create a Sub Main

- Sub Main: Starting point for your application
- Application.Run: Starts the application
- Application.Exit: Quits the application

Practice: Creating a Sub Main



1 Declare module-level variables

2 Create a Sub Main procedure and set it as the startup object

3 Write code for the Selection form

4 Write code to quit the application

5 Test the application

Lesson: Using Predefined Functions

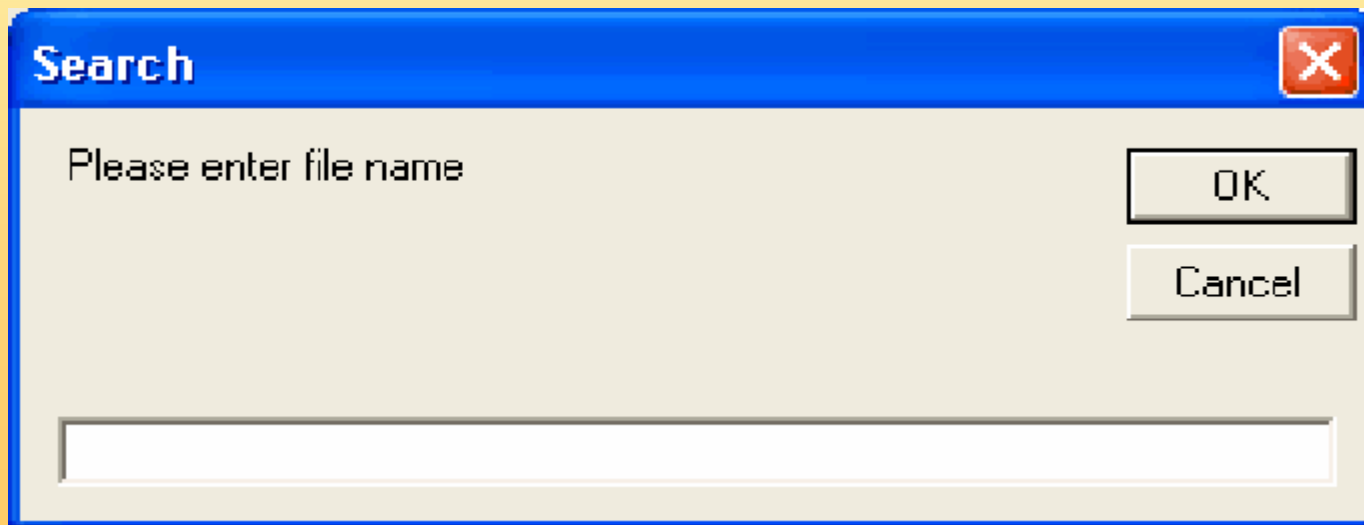
- How to Use the InputBox Function
- How to Use Date and Time Functions
- How to Use String Functions
- How to Use Format Functions
- How to Use Financial Functions

How to Use the InputBox Function

- Displays a prompt in a dialog box, and returns the user input as a string

```
Dim FileName As String
```

```
FileName = InputBox("Please enter file name","Search")
```



How to Use Date and Time Functions

- Perform calculations and operations involving dates and times
- Examples:
 - **DateAdd**: Add or subtract a specific time interval from a date

```
DateAdd(DateInterval.Day, 10, billDate)
```

- **DateDiff**: Determine how many specified time intervals exist between two date/time values

```
DateDiff(DateInterval.Day, Now, secondDate)
```

How to Use String Functions

- Extract only a certain portion of a string
- Return information about a string
- Display information in a particular format
- Examples:

- Trim

```
NewString = Trim(MyString)
```

- Len

```
Length = Len(customerName)
```

- Left

```
Microsoft.VisualBasic.Left(customerName, 5)
```

How to Use Format Functions

- Format numbers, dates, and times according to accepted standards
- Display regional formats without re-coding for nationalities or regions
- Examples:
 - FormatCurrency

```
FormatCurrency(amountOwed, , , TriState.True, TriState.True)
```

- FormatDateTime

```
FormatDateTime(myDate, DateFormat.LongDate)
```

How to Use Financial Functions

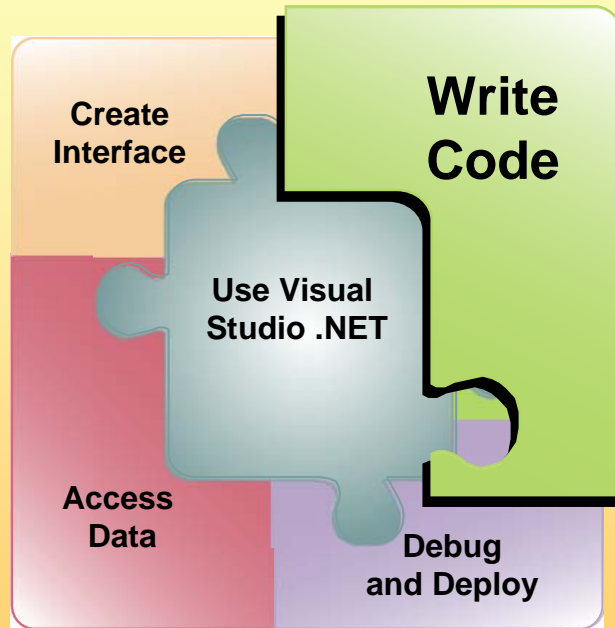
- Perform calculations and operations involving finances; for example, interest rates
- Examples:
 - Pmt

```
payment = Pmt(0.0083, 24, -5000, 0, DueDate.BegOfPeriod)
```

- Rate

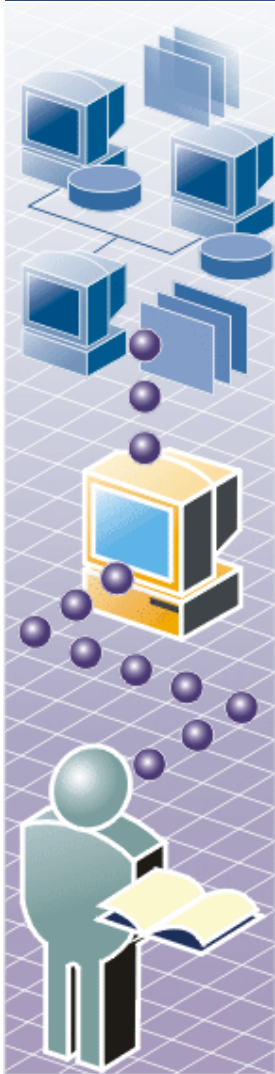
```
ratePerPeriod = Rate(24, 228, -5000, 0, DueDate.BegOfPeriod, _  
0.8)*100
```


Review



- Creating Procedures
- Using Procedures
- Using Predefined Functions

Lab 4.1: Creating and Using Procedures



- Exercise 1: Creating Functions in a Module
- Exercise 2: Working with the Main Form