Macro Programming

Vignesh V
Department of Computer Applications
vigneshv@pes.edu

Macro Programming

Automating Excel with VBA

Vignesh V
Department of Computer Applications

What is VBA

VBA (Visual Basic for Applications) is a programming language built inside Excel. It helps Excel go beyond formulas by:

- Automating repetitive work
- Handling large data tasks quickly
- Reducing manual errors
- Adding features not possible with formulas alone

Example: Instead of manually formatting 50 sales reports, VBA can do it instantly.

Takeaway: VBA turns Excel into a tool that not only calculates but also **acts automatically**.

The Visual Basic Editor (VBE) Interface

The Integrated Development Environment (IDE)

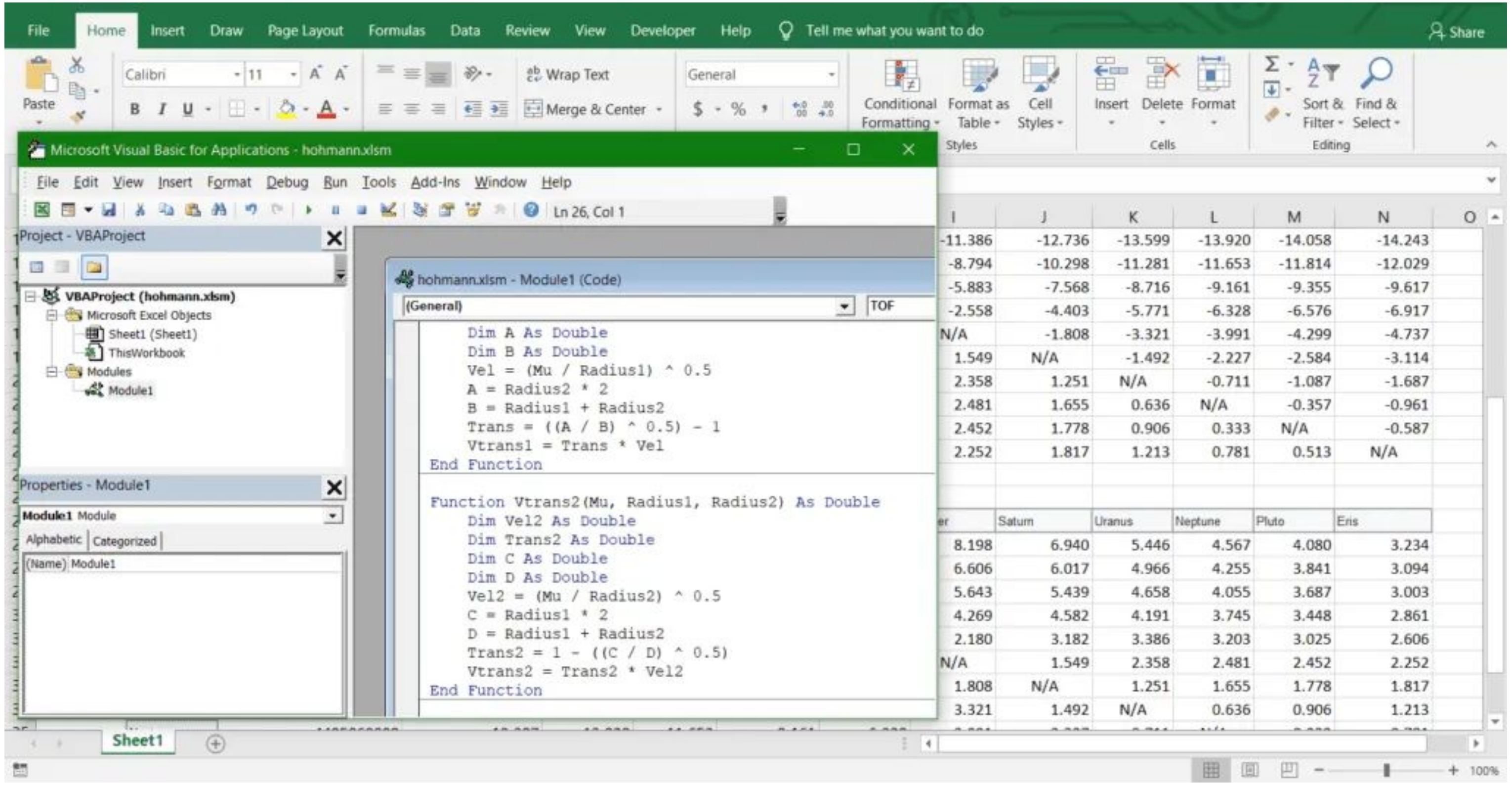
The VBE is accessed via the shortcut **Alt + F11**. It is the environment for all VBA development, including code writing, editing, and debugging.

Primary Components:

1. **Project Explorer (Ctrl+R):** A hierarchical display of all open workbooks and their constituent objects (e.g., worksheets, modules).
2. **Properties Window (F4):** Displays the properties and settings of the currently selected object.
3. **Code Window:** The primary workspace for entering and editing VBA code.

Macro Programming

The Visual Basic Editor (VBE) Interface



```
    Dim A As Double
    Dim B As Double
    Vel = (Mu / Radius1) ^ 0.5
    A = Radius2 * 2
    B = Radius1 + Radius2
    Trans = ((A / B) ^ 0.5) - 1
    Vtrans1 = Trans * Vel
End Function

Function Vtrans2(Mu, Radius1, Radius2) As Double
    Dim Vel2 As Double
    Dim Trans2 As Double
    Dim C As Double
    Dim D As Double
    Vel2 = (Mu / Radius2) ^ 0.5
    C = Radius1 * 2
    D = Radius1 + Radius2
    Trans2 = 1 - ((C / D) ^ 0.5)
    Vtrans2 = Trans2 * Vel2
End Function
```

I	J	K	L	M	N
-11.386	-12.736	-13.599	-13.920	-14.058	-14.243
-8.794	-10.298	-11.281	-11.653	-11.814	-12.029
-5.883	-7.568	-8.716	-9.161	-9.355	-9.617
-2.558	-4.403	-5.771	-6.328	-6.576	-6.917
N/A	-1.808	-3.321	-3.991	-4.299	-4.737
1.549	N/A	-1.492	-2.227	-2.584	-3.114
2.358	1.251	N/A	-0.711	-1.087	-1.687
2.481	1.655	0.636	N/A	-0.357	-0.961
2.452	1.778	0.906	0.333	N/A	-0.587
2.252	1.817	1.213	0.781	0.513	N/A

er	Saturn	Uranus	Neptune	Pluto	Eris
8.198	6.940	5.446	4.567	4.080	3.234
6.606	6.017	4.966	4.255	3.841	3.094
5.643	5.439	4.658	4.055	3.687	3.003
4.269	4.582	4.191	3.745	3.448	2.861
2.180	3.182	3.386	3.203	3.025	2.606
N/A	1.549	2.358	2.481	2.452	2.252
1.808	N/A	1.251	1.655	1.778	1.817
3.321	1.492	N/A	0.636	0.906	1.213

Compiled by Vignesh V

The Visual Basic Editor (VBE) Interface

The Integrated Development Environment (IDE)

The VBE is accessed via the shortcut **Alt + F11**. It is the environment for all VBA development, including code writing, editing, and debugging.

Primary Components:

1. **Project Explorer (Ctrl+R):** A hierarchical display of all open workbooks and their constituent objects (e.g., worksheets, modules).
2. **Properties Window (F4):** Displays the properties and settings of the currently selected object.
3. **Code Window:** The primary workspace for entering and editing VBA code.

Defining Macros

The Principle of Task Automation

A macro is a stored sequence of commands and instructions that can be executed to automate repetitive tasks. It functions as a user-defined procedure to streamline complex or recurring workflows.

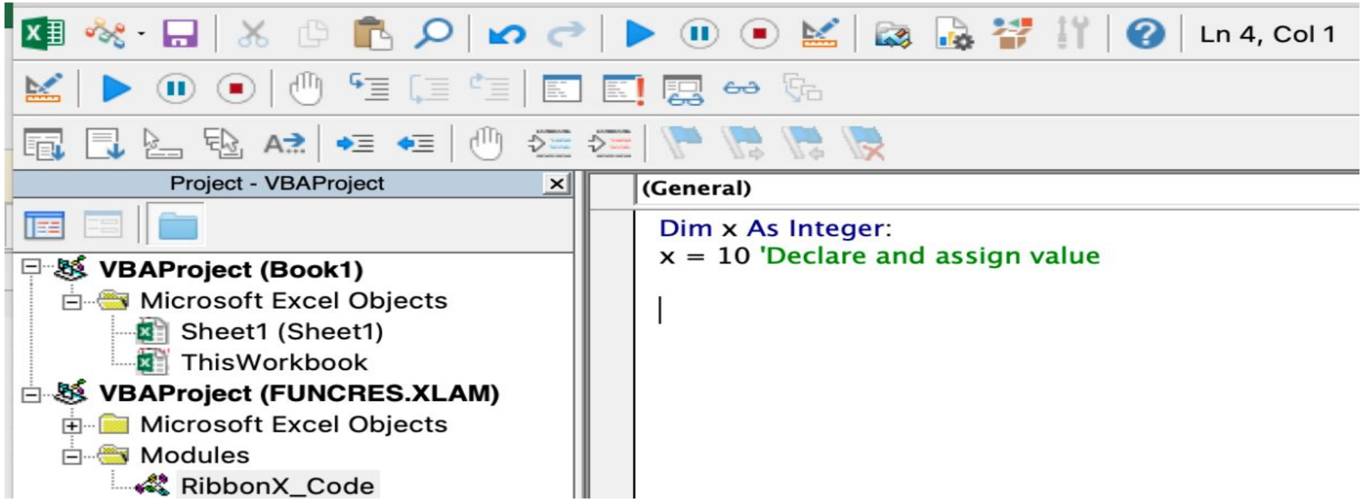
By recording a series of manual actions, developers can create a script that programmatically replicates those actions, enhancing productivity and ensuring procedural consistency.

Defining Macros

- The Principle of Task Automation
- A macro is a stored sequence of commands and instructions that can be executed to automate repetitive tasks. It functions as a user-defined procedure to streamline complex or recurring workflows.
- By recording a series of manual actions, developers can create a script that programmatically replicates those actions, enhancing productivity and ensuring procedural consistency.

VBA Syntax Rules

- **Case-insensitive:** `Range` = `range`.
- **Comments:** Begin with `'` (ignored by Excel).
- **Line Continuation:** Use space + `_` to split long lines.
- **Multiple statements per line:** Separate with `:`.
- **Example:**



```
Dim x As Integer:
x = 10 'Declare and assign value
```

Variables in VBA

A **variable** is a named storage location in memory.

Purpose: **Store values for use in code** so they can be reused, updated, or processed.

Example: Instead of typing `5000` everywhere, store it as `salesAmount`

Common Data Types:

- Integer, Long, Double (numbers).
- String (text).
- Boolean (True/False).
- Variant (any type, flexible but slower).

Variables in VBA

Declaration

General form:

```
Dim variableName As DataType
```

- **Dim** → Tells VBA you are creating a variable.
- **variableName** → Must follow naming rules (no spaces, must start with a letter).
- **DataType** → Defines what type of value the variable can store.

Variables in VBA

Declaration

General form:

```
Dim variableName As DataType
```

- **Dim** → Tells VBA you are creating a variable.
- **variableName** → Must follow naming rules (no spaces, must start with a letter).
- **DataType** → Defines what type of value the variable can store.

Variables in VBA

1. Integer

- **Stores**: Whole numbers.
- **Range**: –32,768 to 32,767.
- **When to use**: Small counts like age, marks, quantity.

Example:

```
 Dim age As Integer
age = 21
```

Variables in VBA

2. Long

- **Stores**: Large whole numbers.
- **Range**: –2,147,483,648 to 2,147,483,647.
- **When to use**: Large counts like population, transaction IDs.

Example:

```
 Dim population As Long
population = 1400000000
```

Variables in VBA

3. Double

- **Stores**: Decimal numbers (floating point).
- **Precision**: Can hold very large/small numbers with decimals.
- **When to use**: Prices, percentages, scientific values.

Example:

```
Dim price As Double
price = 199.95
```

Variables in VBA

4. String

- **Stores**: Text (letters, words, sentences).

- **When to use**: Names, addresses, messages.

Example:

```
 Dim studentName As String
studentName = "Namitha"
```

Variables in VBA

5. Boolean

- **Stores**: Only `True` or `False`.

- **When to use**: Conditions, flags, yes/no questions.

Example:

```
 Dim isEligible As Boolean
isEligible = True
```

Variables in VBA

6. Variant

- **Stores**: Any type of data (number, text, date, etc.).
- **Advantage**: Flexible.
- **Disadvantage**: Slower, uses more memory.
- **When to use**: When the data type is not known in advance.

Variables in VBA

Example:

```
 Dim anything As Variant
anything = "Hello"
anything = 5000
```

Variables in VBA

Best Practice Tip:

- Use the **most specific DataType** possible → faster, more reliable, fewer bugs.

Scope

What is Scope?

- **Scope** = Decides **where in the program a variable can be accessed or used**.

- In VBA, scope depends on **where and how** you declare the variable.

- Local (inside a Sub/Function).
- Module-level (shared within a module).
- Public (shared across the workbook).

Scope

1. Local Scope

- **Definition**: Variable declared **inside a Sub or Function**.
- **Lifetime**: Exists only while that procedure runs.
- **Access**: Can be used **only within that Sub/Function**.

Scope

Example:

```
    Sub CalculateTotal()

        Dim total As Double  'Local variable

        total = 5000

        MsgBox total   'Works here

    End Sub


    Sub ShowTotal()

        MsgBox total   'Error: not visible here
End Sub
```

Scope

2. Module-Level Scope

- **Definition**: Variable declared **at the top of a module**, using `Dim` or `Private`.
- **Lifetime**: Exists as long as the module is in use.
- **Access**: Can be used by **all Subs/Functions inside the same module**.

Scope

Example:

```
    'Declared at top of the module
    Private discount As Double


    Sub SetDiscount()
        discount = 0.1
    End Sub


    Sub ApplyDiscount()
        MsgBox "Discount = " & discount   'Works inside same module
End Sub
```

Scope

3. Public Scope

- **Definition**: Variable declared at the top of a module using `Public`.
- **Lifetime**: Exists as long as the workbook is open.
- **Access**: Can be used **by all modules, Subs, and Functions** in the project.

Scope

Example:

```
'Declared at top of a standard module
Public username As String


Sub SetUser()
    username = "Harry"
End Sub


Sub ShowUser()
    MsgBox "Current user: " & username   'Accessible everywhere
End Sub
```

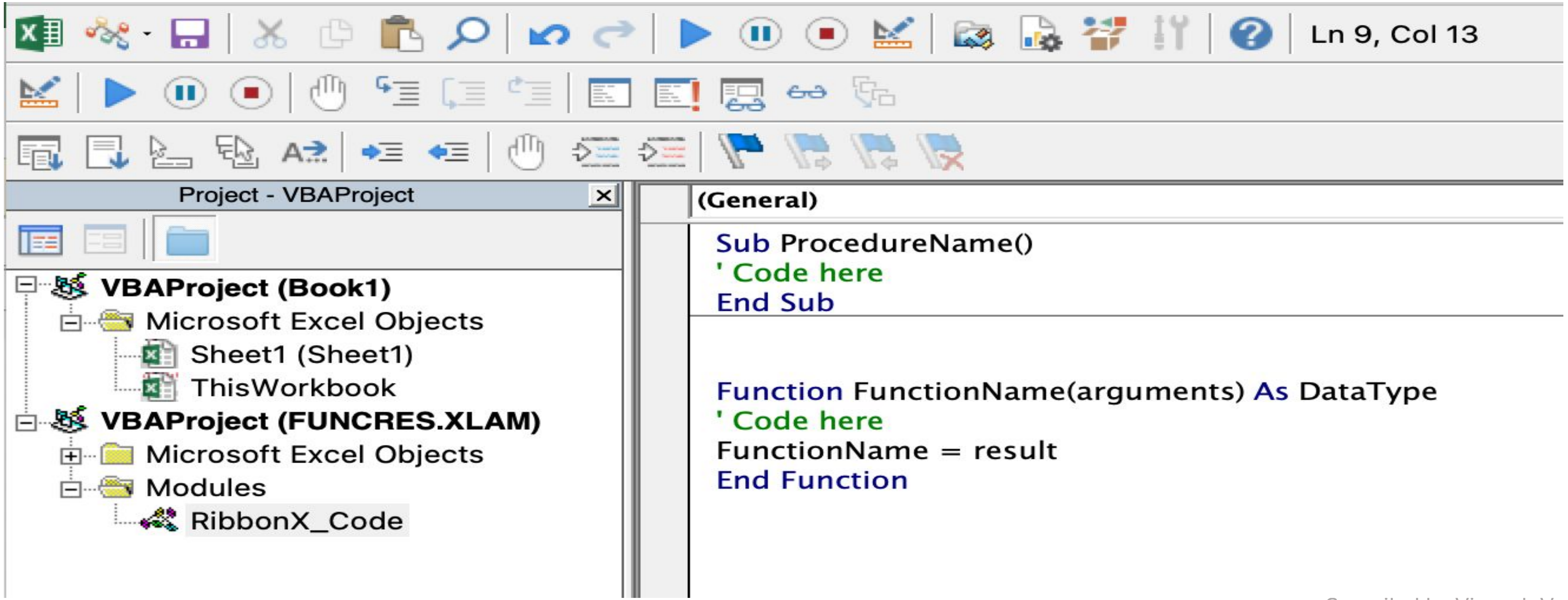
Procedures in VBA

- **Sub Procedures:** Perform actions, no return value.
  Example: Format cells, copy data, display messages.

- **Function Procedures:** Perform calculations and return values.
  Example: Custom formulas in worksheets.

Procedures in VBA

Procedure Structure:

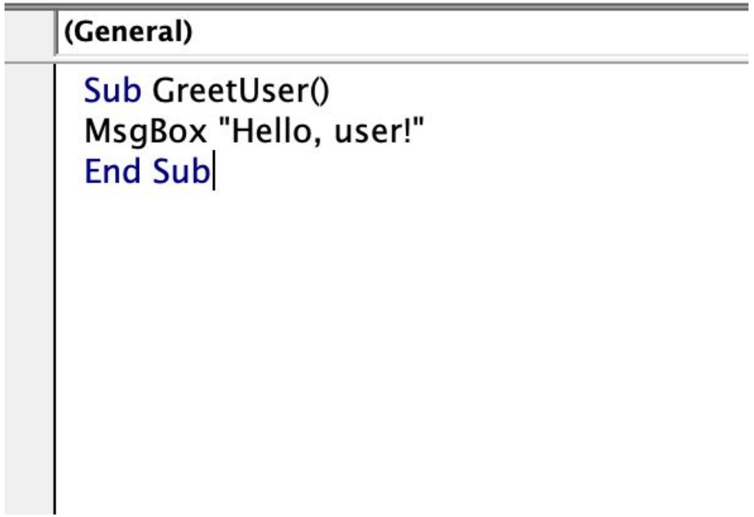


```
Sub ProcedureName()
' Code here
End Sub


Function FunctionName(arguments) As DataType
' Code here
FunctionName = result
End Function
```

Example: Sub Procedure

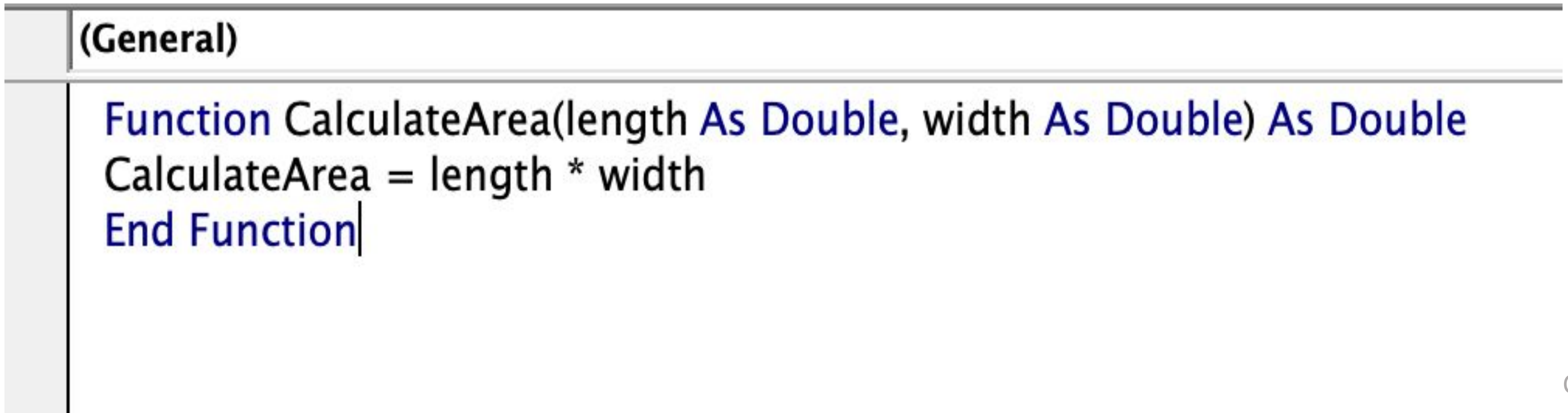
- **Sub GreetUser():** Procedure name.
- **MsgBox:** Displays a message box.
- **End Sub:** Marks the end of the procedure.



```
Sub GreetUser()
MsgBox "Hello, user!"
End Sub
```

Example: Function Procedure

- Takes two inputs (length, width).
- Multiplies them.
- Returns the result.
- Can be used in a worksheet formula: `=CalculateArea(5,10)`.



(General)

```
Function CalculateArea(length As Double, width As Double) As Double
CalculateArea = length * width
End Function
```

Calling Procedures

- **Calling a Sub:**

```
Call GreetUser
' Or simply
GreetUser
```

- **Calling a Function:**

```
Dim area As Double
area = CalculateArea(5, 10)
```

- **In worksheet:** `=CalculateArea(5,10)`

Quiz

Which key combination opens the VBA Editor?

a) Alt + F8

b) Ctrl + F11

c) Alt + F11

d) Ctrl + F8

Which of these is **not** a component of the VBE?

a) Project Explorer

b) Properties Window

c) Formula Bar

d) Code Window

Quiz

Write the syntax to declare a variable named `studentName` as a String

THANK YOU

Vignesh V
Department of Computer Applications
vigneshv@pes.edu