In the end we will conserve only what we love. We love only what we understand. We will understand only what we are taught.. Baba Dioum

From a 1968 speech given at the general assembly of the International Union for Conservation of Nature in Delhi. Mr. Dioum was born in Dahra, Senegal in 1937. He studied ecology at the Ecole Nationale des Eaux et Forêts in Nancy, France where he received a degree in Forestry Engineering.

Why Numerical Methods?

• Many problems in engineering and science cannot be solved in closed (analytical) form
• Still need a solution
• So approximate
• Non-linear problems often require numerical solution

Creating your first VBA code – Your own mini calculator!

• Create two VBA buttons from the developer tab

Program Workspace

Assignment Statements

• \( c = a + b \) means get values of \( a \) and \( b \) from storage locations, add them, store result in location reserved for \( c \)
• Only one variable permitted left of = \( \Rightarrow a + b = c \) not allowed
• But \( c = c + 2 \) \( \Rightarrow \) OK: means take old value of \( c \), add 2 to it, store result in \( c \) location
Simple Calculator

Option Explicit

Sub Codeadd_Click()
REM add values of a and b to produce result c
REM a and b defined within the program
REM answer is displayed on the screen along with
Dim a As Single, b As Single, c As Single
a = Cells(1, 2)
b = Cells(3, 2)
c = a + b
Cells(5, 2) = c
End Sub

Changing the main program name

Name of the routine (Codeadd)

Click properties

Name of the routine (ADD)

Structure of the program

Option Explicit

Sub Codeadd_Click()
REM Code author Dr. Clement
REM add values of a and b to produce result c
REM a and b defined within the program
REM answer is displayed on the screen along with
Dim a As Single, b As Single, c As Single
a = Cells(1, 2)
b = Cells(3, 2)
c = a + b
Cells(5, 2) = c
End Sub

Comment Lines

• VBA and other languages permit adding descriptive notes (non-executable statements) within program
• Specify either by REM (reminder) or an apostrophe (') to include reference text within VBA
• Editor will automatically display the comments in green
• Good idea to label parts of program such as:
  ` REM Computes inverse of a matrix (or)
  ` Computes inverse of a matrix

Documentation

• Important, but often neglected
• Minimum contents:
  Your name
  Purpose of program
  Method
  Input/output including units, if used
  Limitations of program,
  e.g. only finds real roots
• Convenient to use comments at program beginning with REM or ` (separate paper documentation easy to lose)
### Variable Types

- Main variable types used in VBA are:
  - **"Integer"** for integers
  - **"Single"** for decimal numbers (up to 6 significant figures)
  - **"Double"** for decimal numbers (up to 15 significant figures). We will rarely use this.
  - **"String"** for alphanumeric (text) characters

- Define the type of the variables at top of program
- Variables are typed with Dim (short for “dimension”) statement as:
  - Examples
    - Dim x as single
    - Dim i as integer
    - Dim month as string
    - Dim a as single, b as single, c as single
  - *DO NOT DO THIS*
    - Dim a, b, c as single (NOT ACCEPTABLE)

### String variables

- String variables are text variables
- Used for labels, names
- If month typed as string could have
  - `month = "January"

- Two convenient VBA string variables are:
  - **Time$** provides record of when program was executed
  - **Date$** provides current date
- Both useful as documentation for run

### Variable Names

- VBA variables must begin with a letter
- Up to 255 characters long (be descriptive but very long names can be inconvenient)
- No breaks or spaces permitted
- First character must be a letter
- After first character use letters, numbers, _ (underscore), and % e.g. V6x_y18%
- VBA variable names are not case-sensitive; editor will make initial letters the same, based on first appearance of name

### Input/ Output from spreadsheet

- VBA gets input from keyboard, spreadsheet, or a file
- Output can be to screen, spreadsheet, or file
- We will first learn how to input/output from a spreadsheet
- VBA views your spreadsheet as one big array called cells(row, column)

### Interacting with the spreadsheet

- The rows in EXCEL are automatically given a number
- The columns, however, are labeled as A, B, C, D. But you should mentally assign a number to them (as shown below)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>10</td>
<td>11</td>
<td>12</td>
</tr>
</tbody>
</table>

- Cells(5,1) Column number = 1
- Cells(9,3) Row number = 6
The following code shows how to use InputBox to get values and MsgBox to display values.

```vba
Option Explicit
Private Sub CommandButton1_Click()
    Dim a as single, b as single, c as Single
    a = Val(InputBox("Enter the value of A"))
    b = Val(InputBox("Enter the value of B"))
    c = a + b
    MsgBox "Your result is = " & c
End Sub
```

**NOTE, IF VARIABLE IS A STRING THEN INPUT IT AS:**

cityname = InputBox("Enter the name of the city")

---

### Arithmetic operations in VB

- Addition Eg: 2+6 = 8
- Subtraction Eg: 5-3=2
- Multiplication Eg: 2*8 = 16
- Division Eg: 16/2 = 8
- Exponential Eg: 2^5 = 32
- Modulus Eg: 15 Mod 4 = 3

---

### VBA library functions

- **VBA has a limited list of library functions**
- **All angles used or returned in radians**

<table>
<thead>
<tr>
<th>Function</th>
<th>Derived equivalents</th>
</tr>
</thead>
<tbody>
<tr>
<td>Secant(x)</td>
<td>1 / Cos(x)</td>
</tr>
<tr>
<td>Cosecant(x)</td>
<td>1 / Sin(x)</td>
</tr>
<tr>
<td>Cotangent(x)</td>
<td>1 / Tan(x)</td>
</tr>
<tr>
<td>Inverse Sine(x)</td>
<td>Asin(x) / Sqrt(x^2 + 1)</td>
</tr>
<tr>
<td>Inverse Cosine(x)</td>
<td>Arccos(x) / Sqrt(1 - x^2)</td>
</tr>
<tr>
<td>Inverse Hyperbolic Secant</td>
<td>Arcsech(x) + Log[Sqrt(1 - x^2)]</td>
</tr>
<tr>
<td>Inverse Hyperbolic Cosecant</td>
<td>Arcsinh(x) + Log[Sqrt(x^2 + 1)]</td>
</tr>
<tr>
<td>Inverse Hyperbolic Cotangent</td>
<td>Arcctanh(x) + Log[1 / (x - 1)] / 2</td>
</tr>
</tbody>
</table>

- **Logarithm to base N**

  ```vba
  LogN(x) = Log(x) / Log(N)
  ```

---

### Running VBA

- **Multiple ways to run VBA**
- **We will use the following approach**
  - Click on the “Design Mode” under developer tab, this will allow you to switch to the RUN CODE mode
  - Now if you click on the command button your code will execute.
  - If you want to modify the code then click again on the “Design Mode”, which will allow you to toggle between the design and run modes
Dealing with angles

- To convert degrees to radians, multiply degrees by \( \pi/180 \). To convert radians to degrees, multiply radians by \( 180/\pi \).
- If the angle is given as 30 deg, then to compute \( \tan(30) \), use \( \tan(30\times\pi/180) \)

Hierarchy of Operations

- VBA, like most languages, performs operations according to the following hierarchy:
  - raising to a power
  - multiplication/division
  - addition/subtraction
- Operations performed left to right
- Parentheses used to group operations (use them generously)

Hierarchical Operations

- Consider
  
  \[
  \begin{align*}
  x &= 2^3 + 3/4 + 7^8 + 9 - 6 = 8 + 0.75 + 56 + 3 = 67.75 \\
  x &= 6 + 8/2 = 10 \\
  \text{But } x &= (6 + 8)/2 = 7
  \end{align*}
  \]
- Common mistake is not grouping divisors as:
  
  \[
  \begin{align*}
  x &= 2+3/(4+6) = 2.3 \\
  \text{But } x &= 2 + 3/4 + 6 = 8.75
  \end{align*}
  \]

Programming tips:

1. When in doubt, use parentheses and use them generously to avoid mistakes.
2. Split long functions into several small functions using temporary variables.

VBA functions – tips & tricks

Natural log of \( x \) in Excel spreadsheets is \( \ln(x) \); in VB just \( \log(x) \)
Square root of \( x \) in Excel spreadsheets is \( \sqrt{x} \), in VB \( \sqrt{x} \)

- To convert a natural logarithm to base 10, divide the natural logarithm by \( \log(10) \)
- Pi not defined by VBA: Can define with arctangent function as \( \pi = 4\arctan(1) \) or just define \( \pi = 22.0/7.0 \)

Note when you define a number 22.0, VBA displays it as 22\#, this simply means 22.0, is a real number.

Flowcharts

- Provide outline for program like outline for a term paper
- Emphasis on program logic and how various parts fit together
- Flowcharts generic, i.e. independent of programming language used
- Correctly written flowchart makes program easier to write: logic done
- Flowchart separates two areas of difficulty in programming: logic and language syntax. VBA editor helps with syntax errors.
- Basic symbols in flowcharting follow next slide. Symbols connected with arrows show the direction that the program flows.

Types of boxes

- Oval used for starting and ending programs
- Rhombus used for input and output
- Rectangle used for calculations
- Diamond used for decisions
- Hexagon used to delineate loops
- These shapes are just part of a tradition; there are no defined rules, the key is to focus on organizing the logic.
Pseudo code

- Pseudocode is a kind of structured English for describing algorithms.
- Focus on the logic of the algorithm without being distracted by details of language syntax
- Does not use boxes like flowcharts.
- First understand the program specifications
- You must break the main tasks that must be accomplished into smaller ones in order to be able to eventually write a fully developed code.

Pseudo code for simple addition

- Read the first integer
- Read the second integer
- Compute the sum of the two user inputs
- Display the result

Program to Find Real Roots of Quadratic Equation (see Fig 2.11)

- Goal is to solve $ax^2 + bx + c = 0$
- Input $a$, $b$, $c$
- Discriminant, $d = b^2 - 4ac$
- $R1 = \frac{-b + \sqrt{d}}{2a}$
- $R2 = \frac{-b - \sqrt{d}}{2a}$
- Output $R1$ and $R2$

Spreadsheet for Input

Sub Quad4()
' A program to compute the real roots of a quadratic equation, $ax^2 + bx + c = 0$.
' These coefficients are prompted as input from the keyboard. The
' discriminant is calculated and in some cases an error will result in a negative discriminant.
' Execution will halt and an error will result in the square root of a negative discriminant is
' attempted.
' Otherwise the two real roots $r1$ and $r2$ will be computed and output
' in columns A and B.
Dim a as double, b as double, c as double, d as double
Dim r1 as double, r2 as double
Dim X as double, Y as double
a = Cells(1,2)
b = Cells(2,2)
c = Cells(3,2)
d = b*b - 4*a*c
Cells(5,2)= d
r1 = (-b + Sqr(d))/(2*a)
r2 = (-b - Sqr(d))/(2*a)
Cells(6,2)=r1
Cells(7,2)=r2
End Sub

Spreadsheet for Input
Real Root Program Notes

- Brief documentation at beginning of program
- a, b, and c are read from the cells of the spreadsheet
- The discriminant value of \( b^2 - 4ac \) is computed for 2 reasons:
  - sign of discriminant should be checked to see if roots complex (not done in this program. Decisions considered in later chapters)
- avoids evaluating quantity separately in lines for \( r_1 \) and \( r_2 \)
- VBA library function, \( \text{sqr} \), used for square root
- Computes roots, \( r_1 \) and \( r_2 \)
- Check data run with this program: If \( a = 1, b = 2, c = 1 \), \( r_1 = -1 \) and \( r_2 = 1 \)

Output

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>a</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>b</td>
<td>2</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>c</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Discriminant</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>1st Root</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2nd Root</td>
<td>-1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
<td>Roots</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VBA Debugging Tools

VBA has number of tools to assist debugging run-time errors

Good programming practices

- Use generous comments
- Write a legible code that could be understood by someone other than you
- Use parenthesis to separate mathematical operations if the equation is large
- Set up the spreadsheet before you begin coding and use some convention (say, different colors) to differentiate input and output

Debugging

- Programming is fun when everything works but can be frustrating if it does not.
- Difficult to find logic errors in a large program.
- Debugger will help track variables and you can run the program line by line to find out where the error occurs.

Good programming practices

- Do not forget that in programming the value on the right is assigned to left. So Cells(1,1)=a for output and a=Cells(1,1) for input. This is a common mistake.
- Use consistent formatting and separate each code section with a blank line.
- Always have the logic, algorithm or a flowchart of the program ready before you sit in front of the computer to write a program.
- "Run to Cursor" -- program runs to location of cursor without executing that line
- Slowly dragging cursor over variable will display its current value
- Variables not yet calculated labeled Empty
  -- So variables defined in stopped line are Empty
- Can set multiple breakpoints (stopping points) in your program. Click in gray left margin opposite line
  - Breakpoints indicated by brown circle in left margin. Active breakpoints removed by clicking again in left margin. During execution, program stops at each breakpoint.

### Debugging the roots program

- Entered the value of \( a = 0 \) instead of 1 by mistake.

#### Debugging the roots program

- Division by zero is not allowed so we will end up with an error.

#### Debugging the roots program

- Don't know where the error is. So we run the program line by line in debug mode pressing F8..

#### Debugging the roots program

- The line clearly shows we are dividing by zero and if you press F8 again, you will get the overflow error. Therefore, we can identify this as the offending line and fix it.