Computación II
CIA-3351

Unidad I
Introducción a la programación en Visual Basic para Aplicaciones en Excel

Carlos Valle Vidal

Primer Semestre 2018

¿Por qué programar?

- “Programar es lo más cercano que tenemos a un superpoder”

Drew Houston, Creador de Dropbox.

- “Los programadores son los magos del futuro”

Gabe Newell, Director General de Valve.

¿Por qué programar?

- Hoy en día la cantidad de información que hay disponible es demasiada. Procesarla “a mano” es virtualmente imposible, o nos demandaría mucho tiempo.
- El poder “comunicarse” con un computador y darle una secuencia de instrucciones que permita que éste procese toda esta información por nosotros aumentará drásticamente nuestra productividad.

¿Qué aprenderemos en este curso?

- Aprenderemos a procesar información usando Visual Basic para Excel.
- De esta manera seremos capaces de usar las planillas excel para procesar gran cantidad de información automáticamente.
- Para procesar dicha información, tendremos que implementar y diseñar algoritmos usando Visual Basic para Excel.

¿Qué es un algoritmo?

- Un algoritmo es una secuencia de instrucciones bien definidas, ordenadas y finitas que permite llevar a cabo una actividad mediante pasos sucesivos que no generen dudas a quien deba hacer dicha actividad.
- Fueron creados por el matemático Al-Juarismi.
- Los algoritmos presentan estructuras secuenciales, decisionales e iterativas.

Al-Juarismi

Ejemplo: Cocinar arroz

1. Ir a la cocina
2. Sacar el recipiente donde se va cocinar
3. Llenarlo de agua
4. Prender el fogón hasta que se hierva
5. Echar 2 pocillos de agua por 1 de arroz
6. Cuando se seque echar un pocillo de agua con una pizca de sal
7. Para terminar apagar el fogón
8. Servir en platos

Estructura general de un programa

- Para crear nuestro primer programa en Excel vamos a la pestaña Programador->Macros->Editor de Visual Basic
- Si en su Excel no le aparece la pestaña programador:
  1. Haga clic en la pestaña **Archivo**.
  2. Haga clic en **Opciones**.
  3. Haga clic en **Personalizar la cinta de opciones**.
  4. En **Personalizar la cinta de opciones** y **Pestañas principales**, active la casilla **Programador**.
- Si no funciona consulte en la ayuda como **crear una macro**.

Ejercicio: Creando nuestra primera macro

```
Sub Macro1()
'
' Macro1 Macro
' Macro de Prueba
'
' Acceso directo: Ctrl+Mayús+S
'
    Range("A2").Select
    ActiveCell.FormulaR1C1 = "Hola amigos!"
End Sub
```

Declaración de variables

- Una variable es un contenedor que nos permite almacenar información.
- El computador almacena la información en bits (conjuntos de 1's y 0's), por lo que para crear una variable debemos darle un nombre y el tipo de datos correspondiente a la información que almacenaremos en la variable.
- Para declararlas usamos el siguiente formato:

  Dim nombre_variable **As** tipo_dato

- Por ejemplo, si queremos almacenar un número entero en la variable **x:**

  Dim x As Integer

- De esta forma nuestra variable queda bautizada como **x** y puede almacenar un número entero.

Declaración de variables (2)

- Para declarar más variables podemos crearlas en otra línea, o separarlas por coma:
- **Dim x As Integer, y As Integer, z As Integer**
- Pero si todas las variables que queremos crear son del mismo tipo, podemos simplemente hacer:
- **Dim intX, intY, intZ As Integer**

Declaración de variables (3)

Tipo de Dato	Tamaño	Rango
Byte	1 byte	0 to 255
Boolean	2 bytes	**True** or **False**
Integer	2 bytes	-32,768 to 32,767
Long (long integer)	4 bytes	-2,147,483,648 to 2,147,483,647
Single (single-precision floating-point)	4 bytes	-3.402823E38 to -1.401298E-45 para valores negativos; 1.401298E-45 to 3.402823E38 para valores positivos
Double (double-precision floating-point)	8 bytes	-1.79769313486231E308 to -4.940656458412 47E-324 para valores negativos; 4.940656458412 47E-324 to 1.79769313486232E308 para valores positivos

Declaración de variables (3)

- Para declarar variables que contengan texto:

Dim palabra As String

Reglas para los nombres de las variables

- Un nombre de elemento en Visual Basic debe observar las reglas siguientes:
  - Debe comenzar por un carácter alfabético.
  - Sólo puede contener caracteres alfabéticos, dígitos decimales y signos de subrayado (_).
  - No puede superar los 1023 caracteres de longitud.
- Visual Basic no distingue entre variables que solo se diferencian por mayúsculas y minúsculas.
- Por ejemplo, **ABC** y **abc** hacen referencia a la misma variable declarada.

Asignación de variables

- Una vez declarada la variable, almacenamos información en ella en el formato

  nombre_variable = valor

- Por ejemplo, si queremos almacenar un 5 en nuestra variable x:

  X = 5

- Para visualizar el contenido de la variable X:

  ActiveSheet.Range("A1").Value = X

Ejercicio

¿Qué sucede al ejecutar este programa?

```
Sub Macro1()
    Dim x As Integer
    x = 20.6
    Dim y As Double
    y = 20.3
    Range("C3").Value = x
    Range("D3").Value = y
End Sub
```

Asignación de variables (2)

- Para asignar variables de tipo **string** colocamos el contenido entre comillas:

```
palabra = "Hola"
```

- Para visualizar el contenido de la variable X:

```
ActiveSheet.Range("A1").Value = palabra
```

Operadores

- Comúnmente necesitamos hacer operaciones con nuestras variables (sumarlas, multiplicarlas, etc).
- Para esto Visual Basic para Excel nos provee de un conjunto de operadores aritméticos y lógicos.

Operadores Aritméticos

- Entre los operadores aritméticos más conocidos tenemos:
- Suma: +
- Resta: -
- Multiplicación: *
- División: /
- Exponenciación: ^
- Módulo:
- Paréntesis: ()

Suma y Resta

- Suma y resta: +,-

  x + 3

  x - y

- **Dim x, y As Integer**
  y = 10
  x = 20
  Worksheets(1).Range("A1").Value = “Resultado de la suma"
  Worksheets(1).Range("B1").Value = x + y

- Desafío: ¿Cómo ajustar el tamaño de la celda para que se vea toda la frase?

Multiplicación

- Multiplicación: *

  c = a * b

Tenga en cuenta que al igual que en matemáticas la multiplicación tiene precedencia por sobre la suma y la resta.

Ejemplo:

  a = 5 + 4 * 2

Almacenará en la variable a un 13. Si queremos que se realice la suma primero, debemos agregar paréntesis:

 a = (5 + 4) * 2

Almacenará en la variable a un 18.

División

- División: /
- a = b / c
- Almacenará en la variable a el resultado de la división entre b y c. Pero tenga cuidado con el tipo de datos de las variables.
- **Dim a, b, c As Integer**
  b = 3
  c = 2
  a = b / c
- Almacenará en la variable a un 2 en lugar de un 1.5, ya que a es de tipo entero.

División (2)

- El operador división también tiene precedencia sobre la suma y la resta.

Exponenciación

- Exponenciación: ^
- La operación

  a = 2 ^ 3

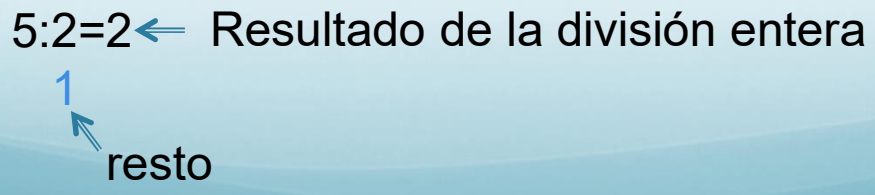
- Almacenará en la variable a el resultado de 2 elevado al cubo.

Operador Módulo

- El módulo retorna el resto de la división entre dos números enteros.
- En Visual Basic para Excel se denota por la palabra reservada **mod**.

  a = 5 mod 2

- Almacena en la variable a el resto de la división entre 5 y 2, es decir 1.



Precedencia de operadores

- Al igual que en matemáticas, los operadores tienen precedencia

1. ()
2. ^
3. *,/
4. +,-

Mayor precedencia

Menor precedencia

Ejemplo

- La instrucción

  a = 3 + 4 * 2 ^ 2

- Realizará la exponenciación primero, luego la multiplicación y finalmente la suma
- Almacenando en la variable a un 19.

Ejercicio

- Escriba un programa que reciba una temperatura en grados Fahrenheit y entregue como resultado el equivalente en grados Celsius.
- La fórmula de conversión es

$$TC = (TF - 32) \times (5/9)$$

- Donde TC es la temperatura en grados Celsius y TF es la temperatura en grados Fahrenheit.

Operadores Relacionales

- Los operadores relacionales se usan para comparar dos valores.

Operador	Nombre	Ejemplo	Significado
<	Menor que	x<y	X es menor que y
>	Mayor que	x>y	X es mayor que y
=	Igual a	x=y	X es igual que y
<>	No igual a	x<>y	X es distinto que y
<=	Menor o igual que	x<=y	X es menor o igual que y
>=	Mayor o igual que	x>=y	X es mayor o igual que y

- Note que el operador **=** sirve también para asignación

Operadores Relacionales

- Una operación relacional devuelve un valor que puede ser TRUE (verdadero) o FALSE (falso)
- Por lo que su resultado se puede almacenar en una variable booleana.
- Ejemplo:

```
Dim a As Boolean

Dim x, y As Integer

x = 5

y = 8

a = x >= y

Worksheets(1).Range("A1").Value = a
```

Operadores Lógicos

- Para combinar expresiones relacionales usamos operadores lógicos.
- AND (Y lógico)

x	y	x AND y
true	true	true
true	false	false
false	true	false
false	false	false

Operadores Lógicos (2)

- OR (O lógico)

x	y	x OR y
true	true	true
true	false	true
false	true	true
false	false	false

- NOT (negación lógica)

x	Not (x)
true	false
false	true

Operadores Lógicos (3)

- Como podemos ver, los operadores lógicos devuelven valores booleanos. Por lo que podemos almacenar su resultado en operadores de este tipo.
- Ejemplo:

```
Dim a As Boolean

Dim x, y As Integer

    x = 5

    y = 8

    a = (x > 3) AND (y <=8)

    Worksheets(1).Range("A1").Value = a
```

Entrada y salida de datos

- Para escribir en una celda de la planilla Excel:

  Hoja.rango_celdas.Value = valor

- Ejemplo:

  Worksheets(1).Range("A1").Value = 20.7

- Esto escribirá 20,7 en la celda A1 de la primera hoja.
- Observe lo que sucede si hacemos:

  Worksheets(1).Range("A1:D3").Value = 20.7

- De esta forma podemos escribir el mismo valor en todo un grupo de celdas.

Instrucción cells

- En lugar de hacer referencia a las celdas de la planilla excel con el rango, podemos usar el comando **Cells** que nos permite hacer referencia a las coordenadas de una celda en particular

```
Dim x As Integer

Worksheets(1).Cells(5, 3) = 20

x = Cells(1,1)

Worksheets(1).Cells(1, 2) = x
```

MsgBox

- Para desplegar mensajes en pantalla, también podemos usar la sentencia **MsgBox:**
- Ejemplos:

1. **MsgBox "Hola a todos!"**
2. **MsgBox “El valor ingresado es" & Range("A1").Value**
3. **MsgBox "Line 1" & vbNewLine & "Line 2"**
4. **Dim x as Integer**

   x = 12

   MsgBox “El valor de x es" & x

- El comando **&** concatena dos strings.

InputBox

- Para almacenar un valor en una variable proveniente desde el teclado podemos usar la sentencia **InputBox**.
- InputBox crea un cuadro de entrada donde el usuario puede ingresar la información que se almacenará en la variable.
- Ejemplo:

```
Dim edad As Integer

edad = InputBox(“Ingrese edad")

MsgBox “Su edad es " & edad
```

InputBox (2)

- También podemos agregarle al botón título y un valor de ingreso **por defecto**.
- Ejemplo:

edad = InputBox(“Ingrese edad", “Cuadro de Ingreso", 20)