

# **Programas de Aplicación III**

## **Tema 2. Elementos del lenguaje (Parte 1)**

Luis Rodríguez Baena y María Dorrego Luxán

**Universidad Pontificia de Salamanca (campus Madrid)**

Facultad de Informática

# Tipos de datos numéricos

Tipo	Ocupación	Comentarios
Long	4 bytes	Valores numéricos enteros
Integer	2 bytes	Valores numéricos enteros
Byte	1 byte	Valores numéricos enteros sin signo.
Single	4 bytes	Valores numéricos reales de simple precisión. Precisión de 7 dígitos
Double	8 bytes	Valores numéricos reales de doble precisión. Precisión de 16 dígitos
Currency	8 bytes	Valores numéricos reales en coma fija, siempre con 4 dígitos decimales.
Decimal	N/A	Valores numéricos de hasta 28 posiciones decimales. Se almacenan en variables de tipo <code>Variant</code> .
Boolean	2 bytes	Valores numéricos. En cualquier valor numérico convertido a <code>Boolean</code> 0 se toma como <code>False</code> y distinto de 0 como <code>True</code> . Un valor <code>Boolean</code> convertido a número toma <code>True</code> como -1 y <code>False</code> como 0.

# Otros tipos de datos

## ❑ El tipo Object.

- Referencias (punteros) a objetos.
- Ocupación: 4 bytes.

## ❑ String

- Cualquier valor no numérico.
- Ocupación:
  - ✓ Cadenas de longitud fija: la señalada en la declaración.
  - ✓ Cadenas de longitud variable: la real más 10 bytes para la longitud.
- Los literales de cadena se separan por comillas dobles.

# Algunas funciones de manipulación de cadenas

Función	Significado
Len( <i>string</i> )	Devuelve el número de caracteres de la cadena <i>string</i>
Mid( <i>string</i> , <i>start</i> [, <i>length</i> ])	Devuelve una subcadena a partir del carácter <i>start</i> de la cadena <i>string</i> de longitud <i>length</i> .
Left( <i>string</i> , <i>length</i> )	Devuelve los <i>length</i> primeros caracteres de la cadena <i>string</i> .
Right( <i>string</i> , <i>length</i> )	Devuelve los <i>length</i> últimos caracteres de la cadena <i>string</i> .
LCase( <i>string</i> )/Ucase( <i>string</i> )	Convierte la cadena <i>string</i> a minúsculas o mayúsculas
InStr([ <i>start</i> ], <i>string1</i> , <i>string2</i> )	Devuelve la posición de la cadena <i>string1</i> dentro de <i>string2</i> a partir de la posición <i>start</i> .
Asc( <i>string</i> )	Devuelve el código del primer carácter de la cadena <i>string</i> .
Chr( <i>código</i> )	Devuelve el carácter correspondiente al valor numérico <i>código</i> .

# El tipo Date (I)

- ❑ Se utiliza para el almacenamiento de fechas y horas.
  - El rango de fechas va desde 1/1/100 hasta el 31/12/9999.
- ❑ Ocupación 8 bytes en coma flotante.
- ❑ Se almacena en forma de números.
  - La parte entera para la fecha.
  - La parte decimal para la hora (.0 para medianoche, .5 para medio día).
- ❑ Literales de fecha.
  - cualquier valor que pueda ser interpretado como fecha y encerrado entre almohadillas.  
#1 de Enero de 2003# #1 Ene 2003# #1-1-2003#

# El tipo Date (II)

## ❑ Admite aritmética de fechas.

- #28-9-2002# + 3 devuelve 1-10-2002.
- Función DateAdd(*intervalo*, *número*, *fecha*)
  - ✓ Añade a la *fecha* el *número* de intervalos señalado.
  - ✓ *intervalo* es una cadena con "yyyy" (año), "q" (trimestre), "m" (mes), "d" (día), "ww", (semana), "h" (hora), "n" (minuto) o "s" (segundo).
  - ✓ DateAdd("ww",3,#28/9/2002#) devuelve 19/10/2002.
- Función DateDiff(*intervalo*, *fecha1*, *fecha2*)
  - ✓ Devuelve el número de intervalos entre dos fechas.
  - ✓ *intervalo* toma los mismos valores que en DateAdd.

# El tipo Date (III)

- Algunas funciones para la manipulación de fechas.

Función	Significado
Date	Devuelve la fecha actual
Now()	Devuelve la fecha y hora actual
Day( <i>fecha</i> )	Devuelve el día del mes
Month( <i>fecha</i> )	Devuelve el mes de la fecha
Year( <i>fecha</i> )	Devuelve el año de la fecha
Weekday( <i>fecha</i> )	Devuelve un entero entre 1 y 7 con el día de la semana. 1 para domingo y 7 para sábado.

# El tipo Variant (I)

- ❑ Comprobación del tipo de dato.
  - Necesaria para saber que operaciones se pueden hacer con un Variant.
  - Funciones de comprobación:
    - ✓ IsNumeric(*variant*)
    - ✓ IsEmpty(*variant*)
    - ✓ IsNull(*variant*)
    - ✓ IsDate(*variant*)
    - ✓ IsArray(*variable*)
    - ✓ IsObject(*variable*)

# El tipo Variant (II)

## ❑ Funciones de conversión

- Fuerzan el tipo de dato almacenado en un Variant.

✓ CBool(*expresión*), CByte(*expresión*), CCur(*expresión*), CDate(*expresión*), CDbl(*expresión*), CDec(*expresión*), CInt(*expresión*), CLng(*expresión*), CSng(*expresión*), CStr(*expresión*), CVar(*expresión*).

## ❑ Función VarType(*expresión*) devuelve el tipo de dato de un variant.

Constante simbólica	Valor	Significado	Constante simbólica	Valor	Significado
vbEmpty	0	Vacía	vbString	8	Cadena
vbNull	1	Null	vbObject	9	Objeto
vbInteger	2	Entero	vbBoolean	11	Boolean
vbLong	3	Entero Largo	vbVariant	12	Variant (Array)
vbSingle	4	Simple	vbDecimal	14	Valor decimal
vbDouble	5	Doble	vbByte	17	Byte
vbCurrency	6	Currency	vbArray	8192	Array
vbDate	7	Fecha			

# Identificadores

- ❑ Logitud máxima: 255 caracteres.
- ❑ Reglas de formación:
  - Compuestas por cualquier carácter alfabético del juego de caracteres.
  - Puede contener dígitos y el carácter de subrayado.
  - Debe comenzar por un carácter alfabético.
  - Debe ser único para objetos en el mismo ámbito.
  - No es sensible a mayúsculas.

# Constantes

- ❑ Constantes literales.
- ❑ Constantes intrínsecas o definidas por el sistema.
  - De las bibliotecas de VisualBasic y VBA (comienzan por vb)
    - ✓ vbRed, vb
  - De controles y bibliotecas de objetos.
- ❑ Constantes simbólicas.

`[Public|Private] Const identificador [As tipo] = expresión`

# Variables (I)

## ❑ Declaración implícita.

- Cualquier variable se declara por el mero hecho de utilizarla.

## ❑ Declaración explícita.

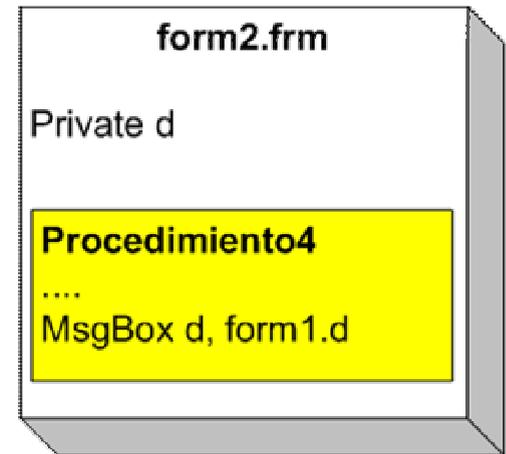
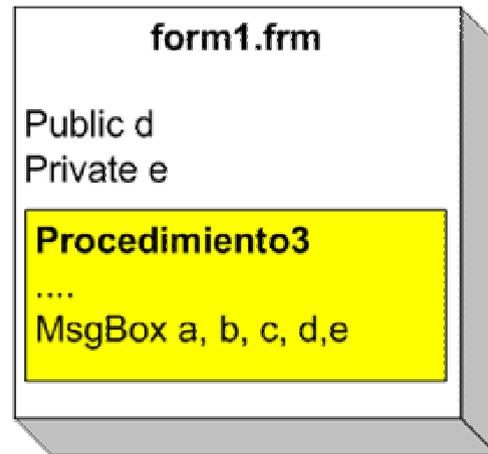
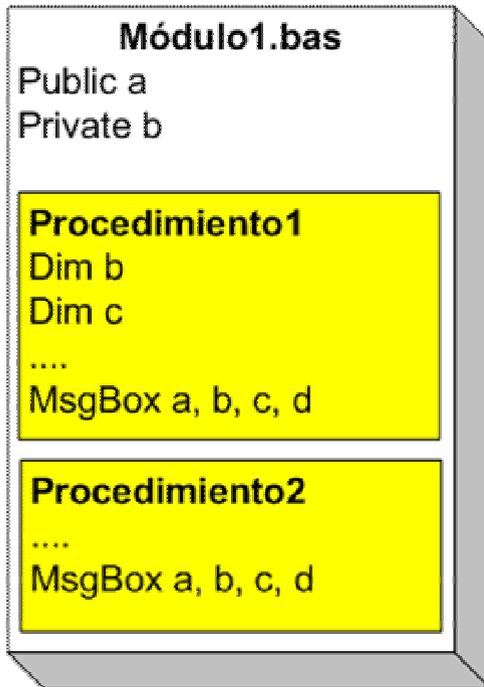
```
[Public|Private|Static|Dim] NomVar [As [New] TipoDato]
```



```
[, NomVar [As [New] TipoDato]...
```

- Public, Private, Dim indican el ámbito de la variable.
- Static indica el ámbito y el tipo de vida de la variable.
- *TipoDato* puede ser cualquier tipo de dato estándar, perteneciente a una biblioteca de clases o definido por el usuario.
- El tipo de dato es opcional, por omisión se trata de variables de tipo Variant.
- New crea una nueva instancia de una clase.
- La declaración `Option Explicit` colocada en la sección de declaraciones de un módulo obliga a declarar todas las variables de forma explícita.

# Variables (II)



# Operadores aritméticos

- ❑ Suma / resta (+, -)
- ❑ Multiplicación / división real (\*, /)
- ❑ Exponenciación (^).
- ❑ Resto de la división entera (mod).
- ❑ División entera (\).
- ❑ Negación (-).
- ❑ El resultado de la operación depende de los operandos y los operadores.
  - Como norma será el de la expresión con más precisión.
  - En la división real y la exponenciación, el resultado es real.

# Operadores de relación y lógicos

## ❑ Operadores de relación.

- Operan con datos del mismo tipo y devuelven un dato de tipo lógico.
- =, <, >, <=, >=, <>.
- Cualquier comparación con un dato de tipo `Null` devuelve `Null`.
  - ✓ `a = Null`, devuelve `Null`.
  - ✓ `IsNull(a)`, devuelve `True`.

## ❑ Operadores lógicos.

- Operan sobre datos de tipo lógico y devuelven un dato de tipo lógico.
- AND, OR, NOT, XOR,

# Operadores de cadena

## ❑ Concatenación: + y &.

- El operador + puede sumar o concatenar.
  - ✓ Si los dos datos son numéricos o Variant de tipo numérico, suma.
  - ✓ Si los dos datos son de cadena o Variant de tipo cadena, concatena.
  - ✓ Si un dato es de tipo cadena y otro cualquier Variant excepto Null, concatena.
- El operador & siempre concatena.

## ❑ Operador Like

- Compara una cadena con un patrón.

*expresiónCadena Like patrón*

- Patrón es una cadena que puede contener:
  - ✓ \*, sustituye a 0 o más caracteres ("hola" Like "ho\*").
  - ✓ ?, sustituye a 1 carácter ("hola" Like "ho?a").
  - ✓ #, sustituye a un dígito ("1234A" Like "#####").
  - ✓ [caracteres], sustituye a alguno de los caracteres ("hola" Like "hol[aeiou]").
  - ✓ [car1-car2], sustituye a alguno de los caracteres del rango ("A3" Like "A[0-5]").
  - ✓ [!caracteres], sustituye a cualquier carácter no incluido en la lista.

# Comparaciones de cadenas

## ❑ Declaración `Option Compare {Binary | Text}`.

- Aparece en la sección de declaraciones de un módulo.
  - ✓ Binary (opción por omisión), realiza la comparación utilizando el código de los caracteres.
  - ✓ Text, realiza la comparación utilizando la configuración regional de Windows.

## ❑ Función `StrComp (cad1, cad2 [, opc])`.

- Devuelve 0 si cad1 y cad2 son iguales, 1 si cad1>cad2 y -1 si cad1<cad2.
- opc, indica el tipo de comparación:
  - ✓ -1 (vbUseCompareOption), utiliza lo definido en Option Compare
  - ✓ 0 (vbBinaryCompare), utiliza la comparación binaria.
  - ✓ 1 (vbTextCompare), utiliza la comparación de texto.

# Operadores de objetos

- ❑ Comparación de referencias de objetos: `Is`.
  - Comprueba si dos variables de tipo objeto referencia al mismo objeto.
  
- ❑ Operador `TypeOf`.
  - Determina la clase a la que pertenece un objeto.
  - Se usa en combinación con `Is`
    - ? `TypeOf a = CommandButton`

# Prioridad de los operadores

+  -



Aritméticos	Relación	Lógicos
Exponenciación (^)	Igualdad (=)	No ( <b>Not</b> )
Negación (-)	Desigualdad (<>)	Y ( <b>And</b> )
Multiplicación y división (* /)	Menor que (<)	O ( <b>Or</b> )
División entera (\)	Mayor que (>)	O exclusivo ( <b>Xor</b> )
Módulo ( <b>Mod</b> )	Menor o igual que (<=)	Equivalencia ( <b>Eqv</b> )
Suma y resta (+ -)	Mayor o igual que (>=)	Implicación ( <b>Imp</b> )
Concatenación de cadenas (& +)	<b>Like</b>	
	<b>Is</b>	

# Operadores de asignación

## ❑ Asignación de variables: Let.

`[Let] variable = expresión`

- Sólo se pueden asignar tipos compatibles.
- A un dato Variant puede asignarse cualquier otro dato, pero no siempre al revés.
- Un dato Variant distinto de Null puede ser asignado a cualquier cadena.
- Un dato Variant distinto de Null puede ser asignado a cualquier dato numérico si IsNumeric es verdadero.

## ❑ Asignación de objetos: Set

`Set variableObjeto = {[New] expresiónObjeto | Nothing}`

- New crea una nueva instancia del objeto.
- Nothing destruye la referencia al objeto.

# Estructuras selectivas (I)

## ❑ Declaración If.

```
If expresiónLógica Then instrucción [Else instrucción]
```

## ❑ Declaración If de bloques.

```
If expresiónLógica Then  
    instrucciones
```

```
[ElseIf expresiónLógica Then  
    instrucciones]...
```

```
[Else  
    instrucciones]
```

```
End If
```

# Estructuras selectivas (II)

## ❑ Declaración Select Case.

```
Select Case expresiónPrueba  
  [Case listaDePruebas  
    [instrucciones]] ...  
  [Case Else  
    [instrucciones]]
```

- *listaDePruebas* puede ser:
  - ✓ Una expresión.
  - ✓ Dos expresiones separadas por la palabra reservada `TO`.
  - ✓ `Is` *operadorDeRelación* *expresión*.
  - ✓ Una mezcla de todo lo anterior separados por comas.
- La declaración `Select Case` hace cortocircuito.

# Otras funciones de decisión

## ❑ Función `IIf`.

`IIf(expresiónLógica, expr1, expr2)`

- Si la expresión es verdadera devuelve `valor1`, si no `valor2`.

## ❑ Función `Switch`.

`Switch(exprLog1, expr1[, exprLog2, expr2]...)`

- Evalúa cada expresión y si es verdadera devuelve el valor asociado.
- Devuelve `Null` si ninguna de las expresiones es verdadera.

## ❑ Función `Choose`.

`Choose(índice, expr1 [, expr2]...)`

- `índice` es un valor entre 1 y n.
- Si `índice` es 1 devuelve `expr1`, si es 2 `expr2`...
- Si es `índice` menor que 1 o mayor que el número de expresiones devuelve `Null`.

# Estructuras repetitivas (I)

## □ Declaración Do...Loop.

```
Do [{While | Until} exprLógica]  
  [instrucciones]  
  [Exit Do]  
  [instrucciones]
```

Loop

```
Do  
  [instrucciones]  
  [Exit Do]  
  [instrucciones]  
Loop [{While | Until} exprLógica]
```

# Estructuras repetitivas (II)

## ❑ Declaración While..Wend.

```
While expresiónLógica  
    [instrucciones]  
Wend
```

## ❑ Declaración For..Next.

```
For variable = exprInicio To exprFin [Step exprIncremento]  
    [instrucciones]  
    [Exit For]  
Next [variable]
```

# Estructuras repetitivas (III)

## ❑ Declaración For..Each.

```
For Each elemento In grupo  
    [instrucciones]  
    [Exit For]  
    [instrucciones]  
Next [elemento]
```

- Repite las instrucciones por cada elemento de una colección o array.
- *grupo* es una colección o array.
  - ✓ Una colección es objeto que contiene una serie de elementos relacionados (objetos, controles, datos...).
- *elemento* es una variable que va tomando el valor de cada uno de los componentes del grupo.
- Ejemplo:

```
Dim ctl as Object  
For Each ctl In Controls  
    MsgBox ctl.Name  
Next
```

# Declaración With

- ❑ Ejecuta una serie de instrucciones sobre un único objeto o tipo definido por el usuario.

```
With objeto
```

```
    [instrucciones]
```

```
End With
```

- Para referirnos al objeto se utiliza el punto.

- Ejemplo

```
With Label1
```

```
    .Height = 2000
```

```
    .Width = 2000
```

```
    .Caption = "Una etiqueta"
```

```
End With
```