Microsoft® Visual Studio®

Visual Basic

Introducción a la Programación con Visual Basic

Pedro Corcuera

Dpto. Matemática Aplicada y Ciencias de la Computación

Universidad de Cantabria

corcuerp@unican.es

Objetivos

- Presentar los elementos necesarios para la programación en el lenguaje Visual Basic utilizando el entorno Visual Studio

Historia

- BASIC stands for Beginner's All-purpose Symbolic Instruction Code.
- BASIC was developed by John Kemeny and Thomas Kurtz at Dartmouth in mid-1960s.
- Visual Basic was released in 1991 by the Microsoft Corporation.
- Visual Basic 2012 is similar to original Visual Basic, but more powerful

Introducción

- A language used to create windows applications.
- Programs are developed with a Graphical User Interface or GUI.
- The instructions executed in the program are controlled by events.

Vista vs Windows 7 & 8

Vista

Windows 7

Windows 8

Dónde encontrar VB de forma libre (gratis)

Visual Studio en Microsoft DreamSpark (SdeI - UC):

https://sdei.unican.es/Paginas/servicios/software/DreamSpark.aspx

Información oficial:

- Microsoft Developer Network (MSDN):

http://msdn.microsoft.com/es-es/library/2x7h1hfk.aspx

Terminología

A computer program may also be called:

- Project
- Application
- Solution
- Elements:
  - Form
  - Control
  - Object
  - Properties
  - Event

Ciclo de desarrollo de un programa

- The programmer has to know how to solve problems
  - Determine *Output*
  - Identify *Input*
  - Determine *process* necessary to turn given *Input* into desired *Output*

1. Analyze: Define the problem.
2. Design: Plan the solution to the problem.
3. Choose the interface: Select the objects (text boxes, buttons, etc.).

Ciclo de desarrollo de un programa

4. Code: Translate the algorithm into a programming language.
5. Test and debug: Locate and remove any errors in the program.
6. Complete the documentation: Organize all the materials that describe the program.

Herramientas de programación

Three tools are used to convert *algorithms* into computer programs:

- *Flowchart* - Graphically depicts the logical steps to carry out a task and shows how the steps relate to each other.
- *Pseudocode* - Uses English-like phrases with some Visual Basic terms to outline the program.
- *Hierarchy chart* - Shows how the different parts of a program relate to each other.

Algoritmo

A step-by-step series of instructions for solving a problem (a recipe is an example of an algorithm).

Visual Basic

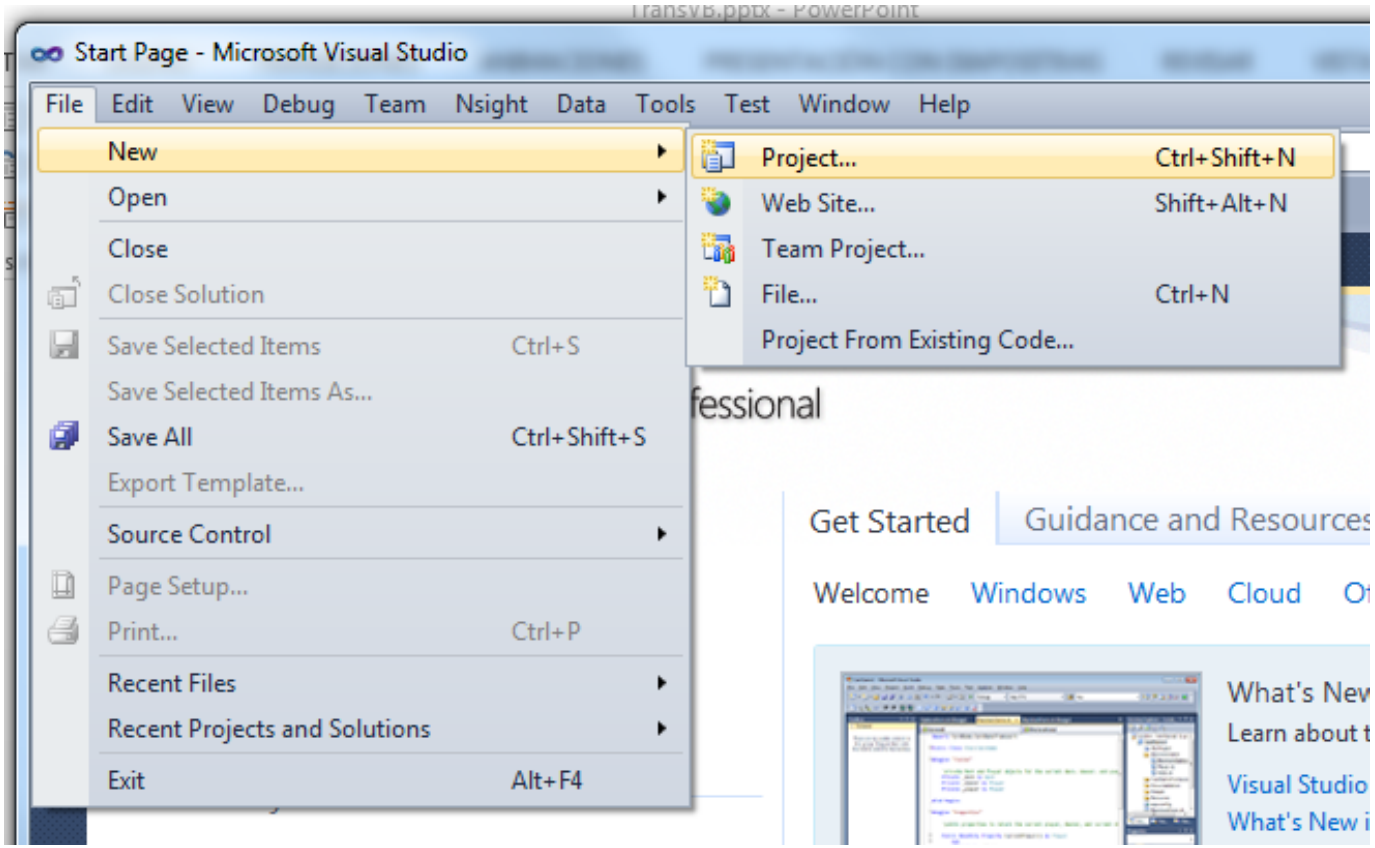
- Language used to create Windows applications.
- Provides a Graphical User Interface or GUI.
- The sequence of instructions executed in the program is controlled by events.

Cómo desarrollar una aplicación VB

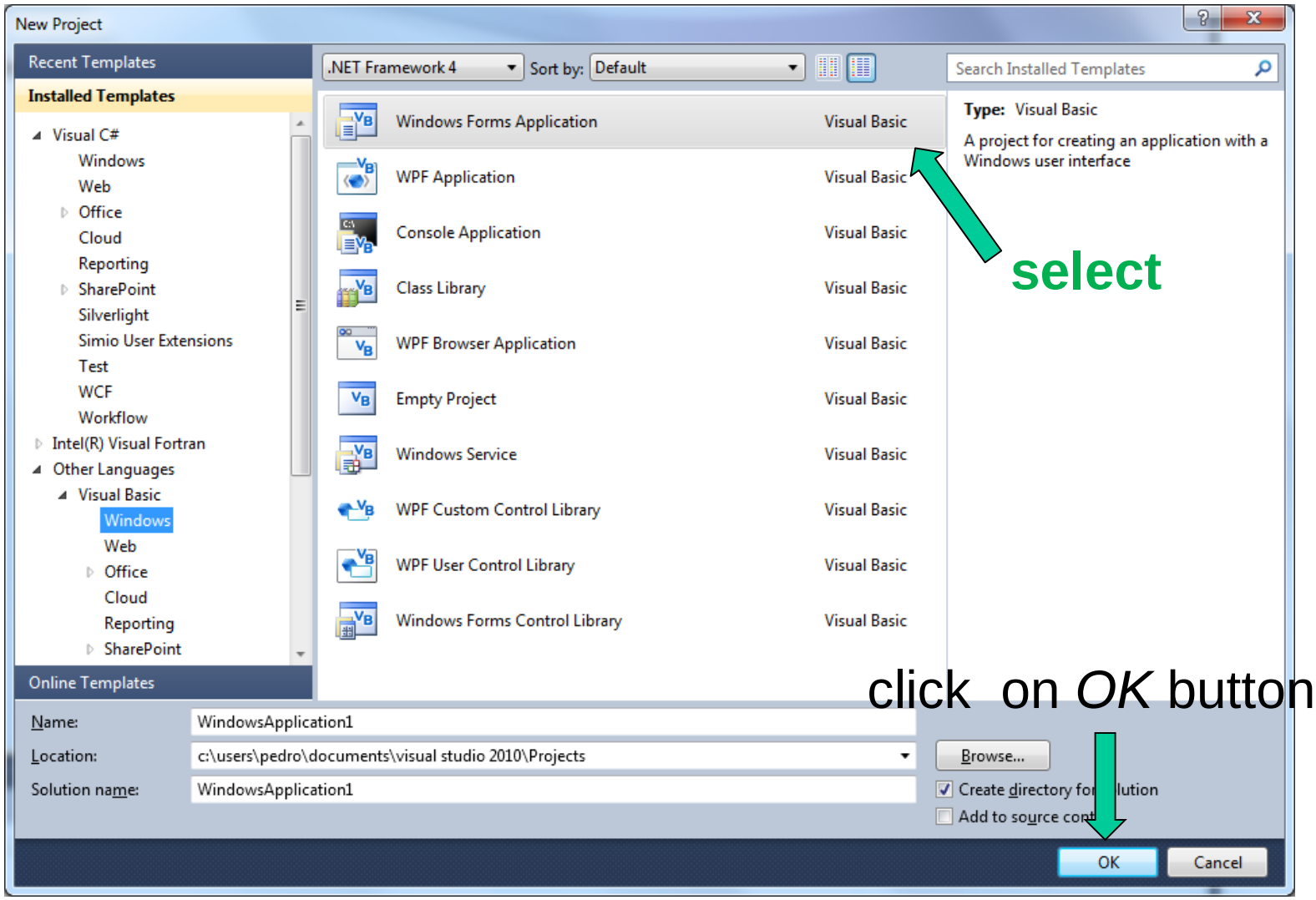
- Design the Interface for the user.
- Determine which events the controls on the window should recognize.
- Write the event procedures for those events.

Desarrollo de un programa VB

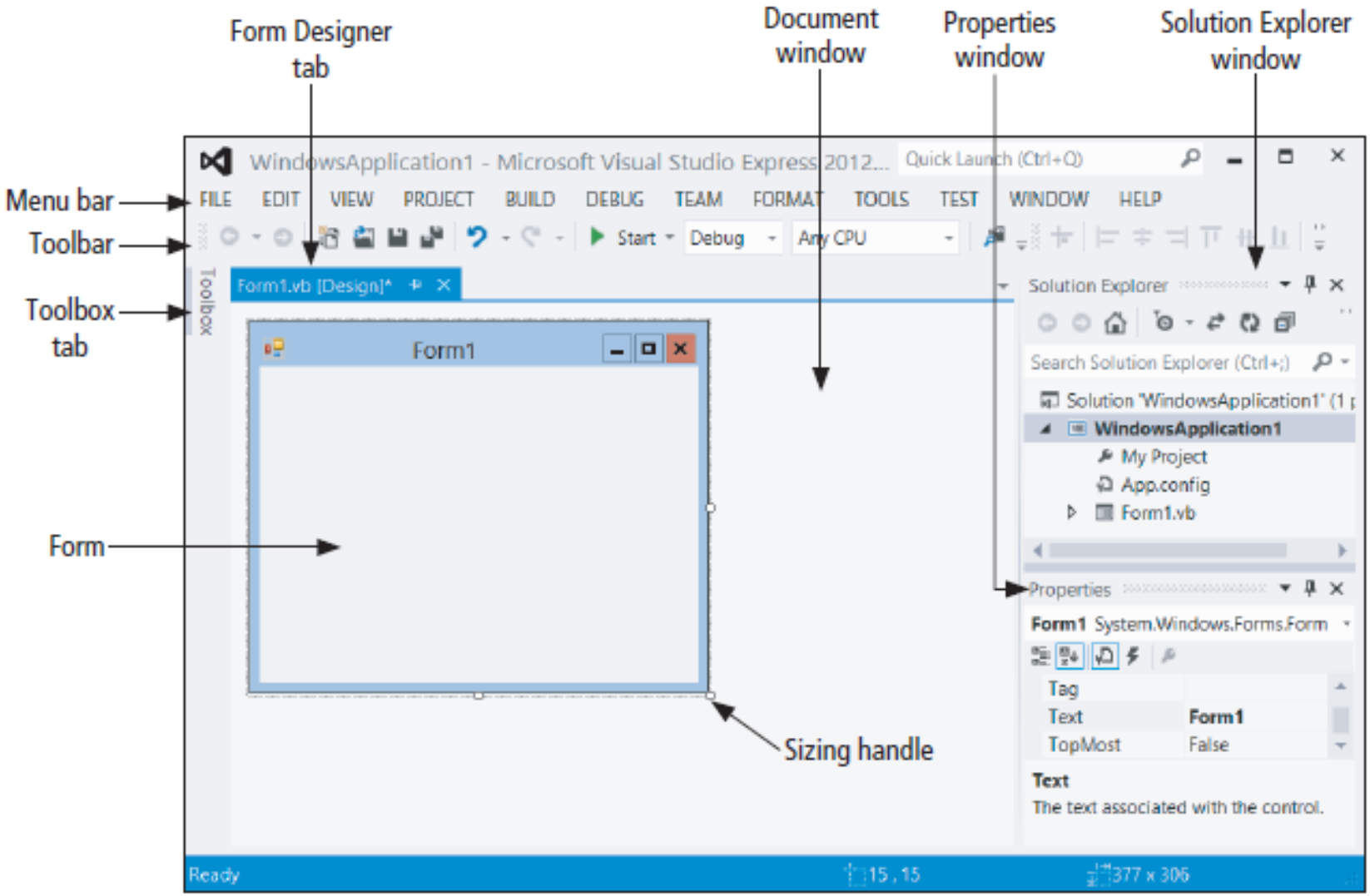
- Iniciar el programa Microsoft Visual Studio
- Crear un proyecto



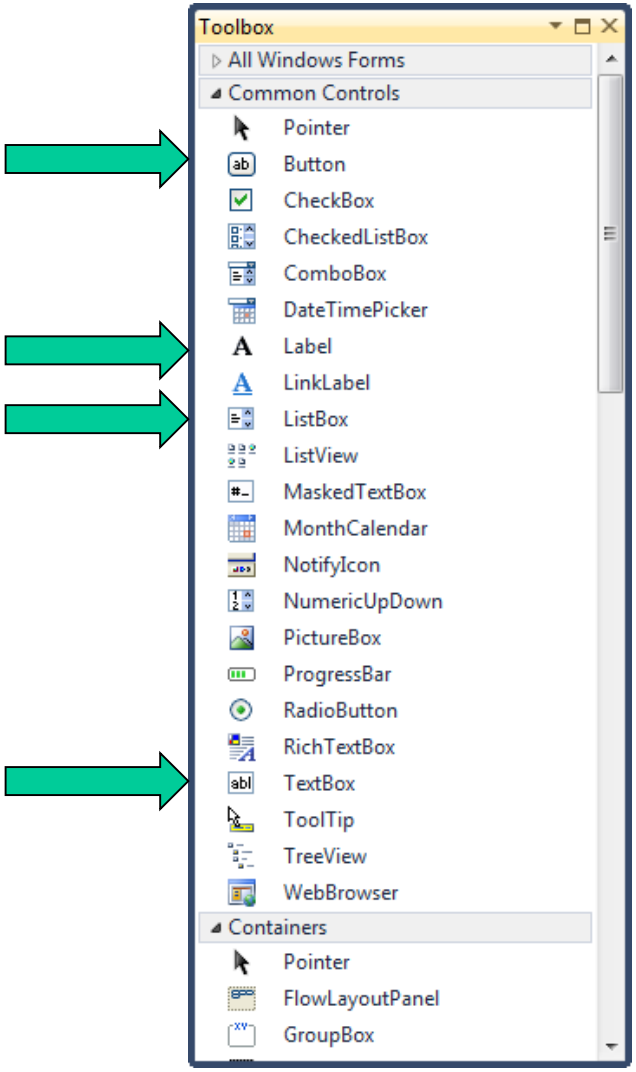
Crear proyecto



Pantalla inicial VB



Toolbox

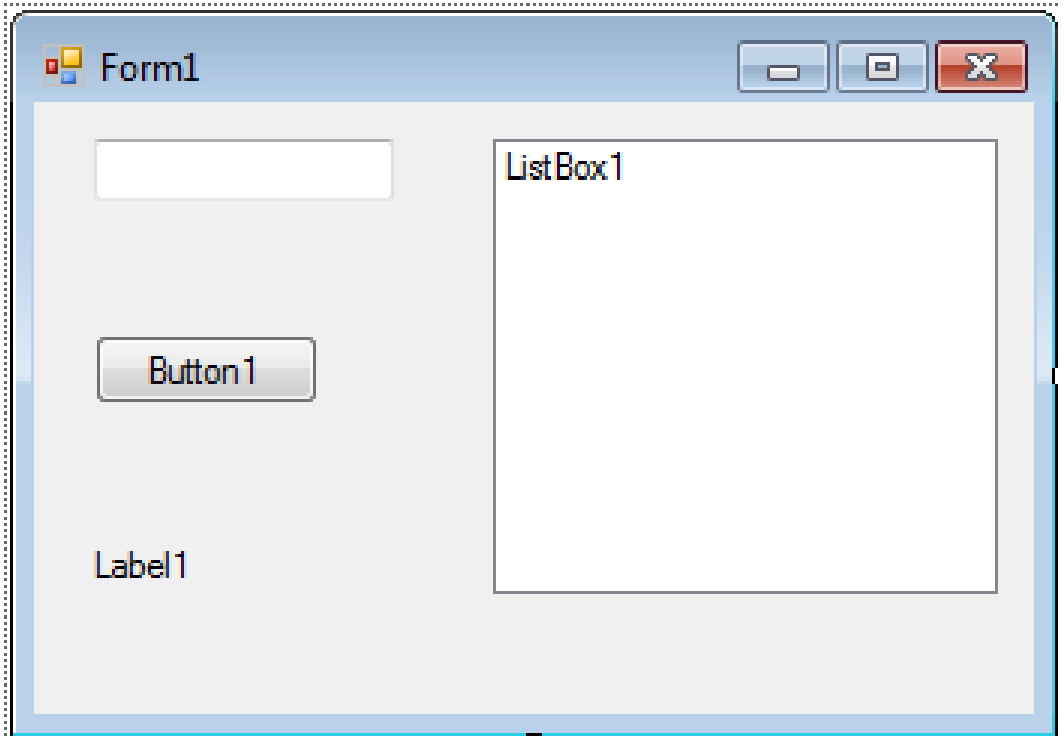

Toolbox
All Windows Forms
Common Controls
Pointer
Button
CheckBox
CheckedListBox
ComboBox
DateTimePicker
Label
LinkLabel
ListBox
ListView
MaskedTextBox
MonthCalendar
NotifyIcon
NumericUpDown
PictureBox
ProgressBar
RadioButton
RichTextBox
TextBox
ToolTip
TreeView
WebBrowser
Containers
Pointer
FlowLayoutPanel
GroupBox

Formas de poner un control en un Form

- Double-click
- Drag and Drop
- Click, Point, and Click
- Click, Point, and Drag

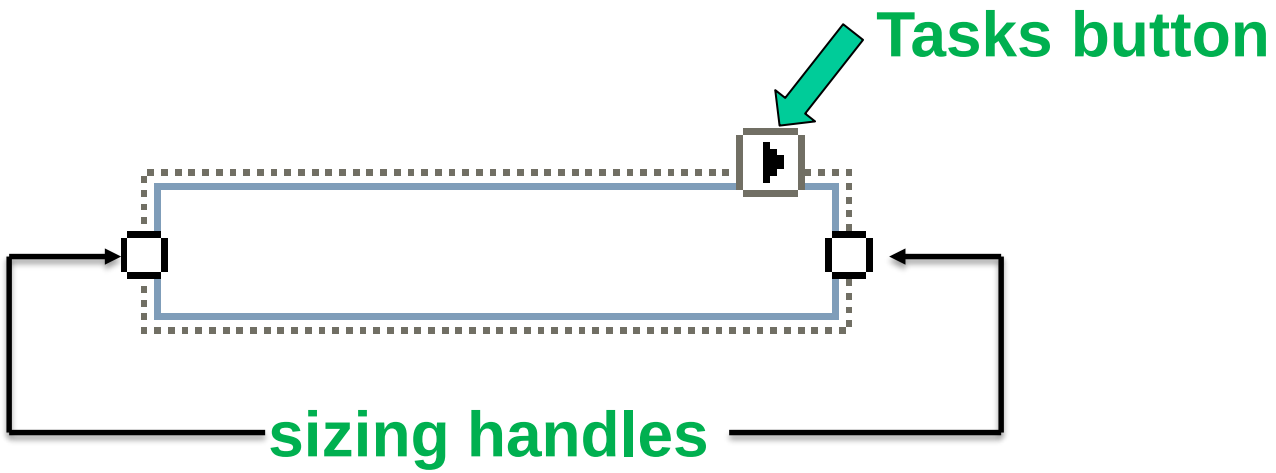
Controles en tiempo de diseño

- To select a control, click on it. Sizing handles appear when a control is selected.

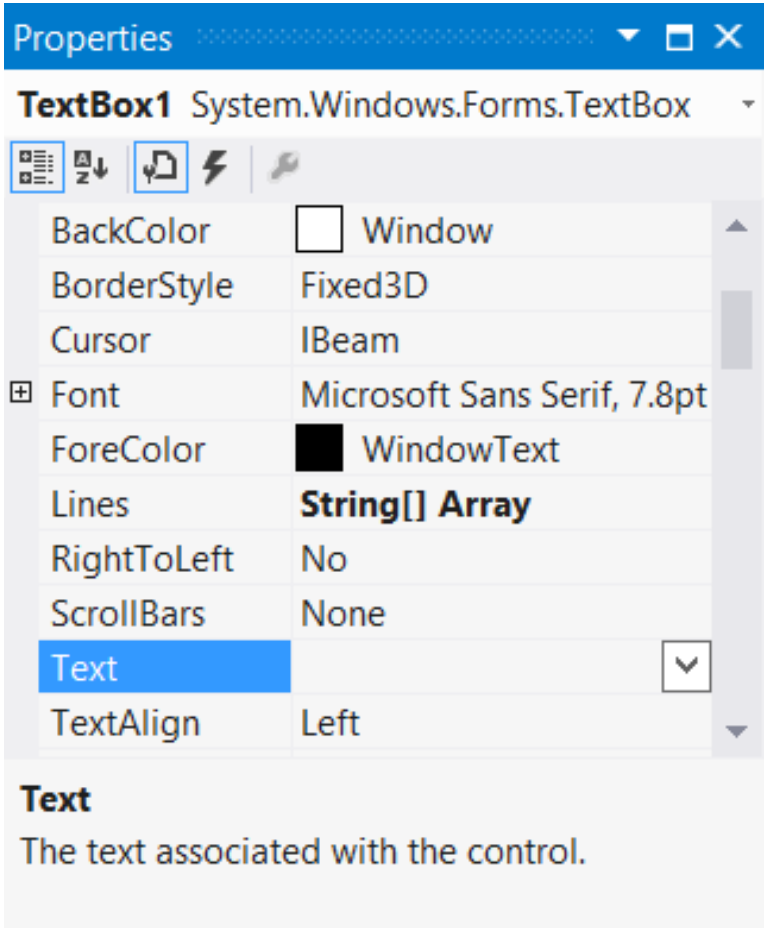


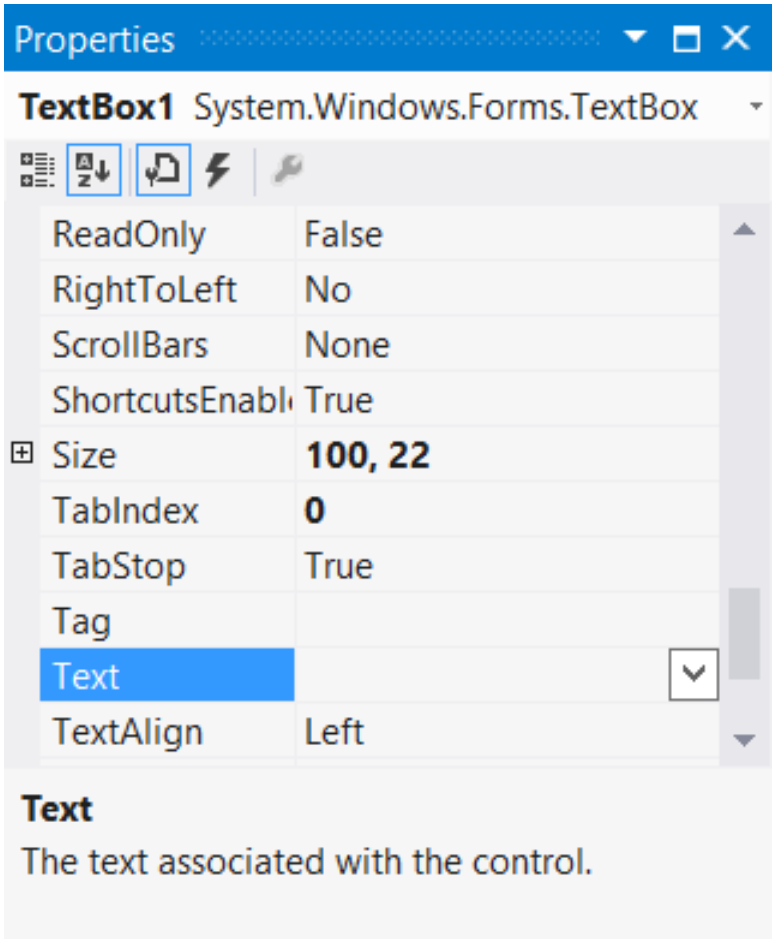
Text Box Control

- Used for input and output
- When used for output, ReadOnly property is set to True



Ventana de Propiedades



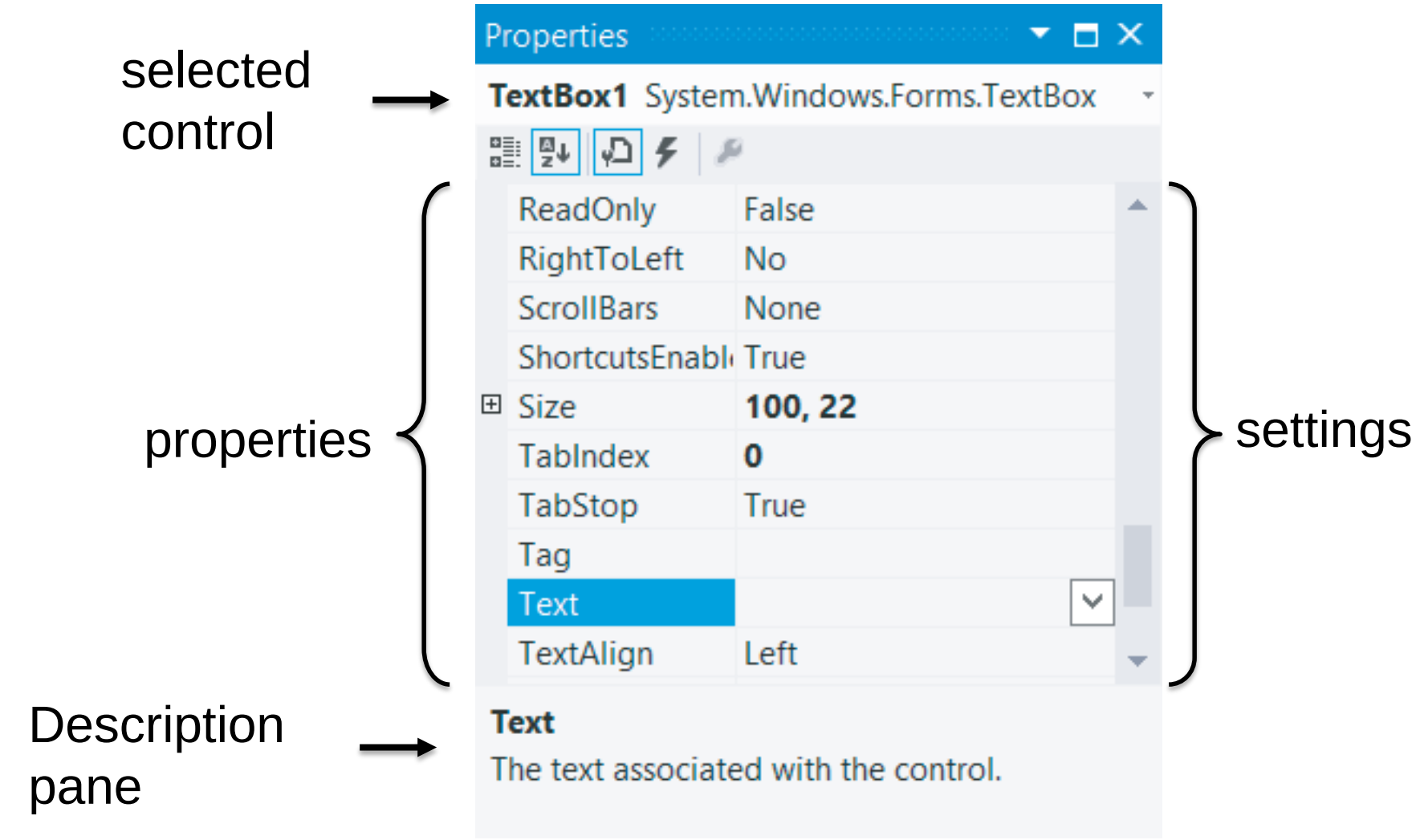


Press F4 to display the Properties window for the selected control.

categorized view

alphabetical view

Ventana de Propiedades



22

Propiedades más usadas

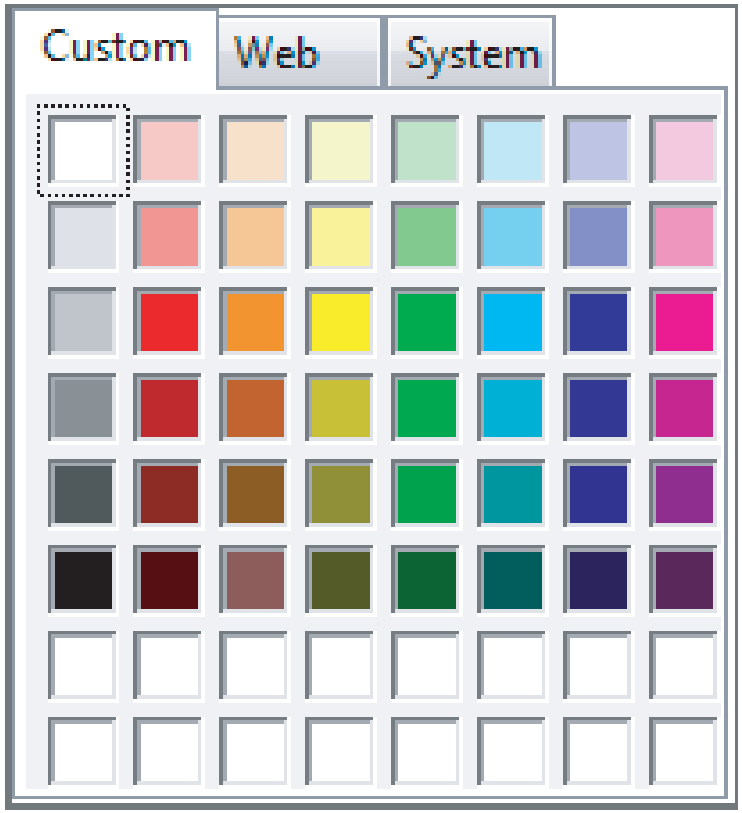
- Text
- Autosize
- Font.Name
- Font.Size
- ForeColor
- BackColor
- ReadOnly

Asignación de propiedades

- Click on property name in left column.
- Enter its setting into right column by typing or selecting from options displayed via a button or ellipsis.

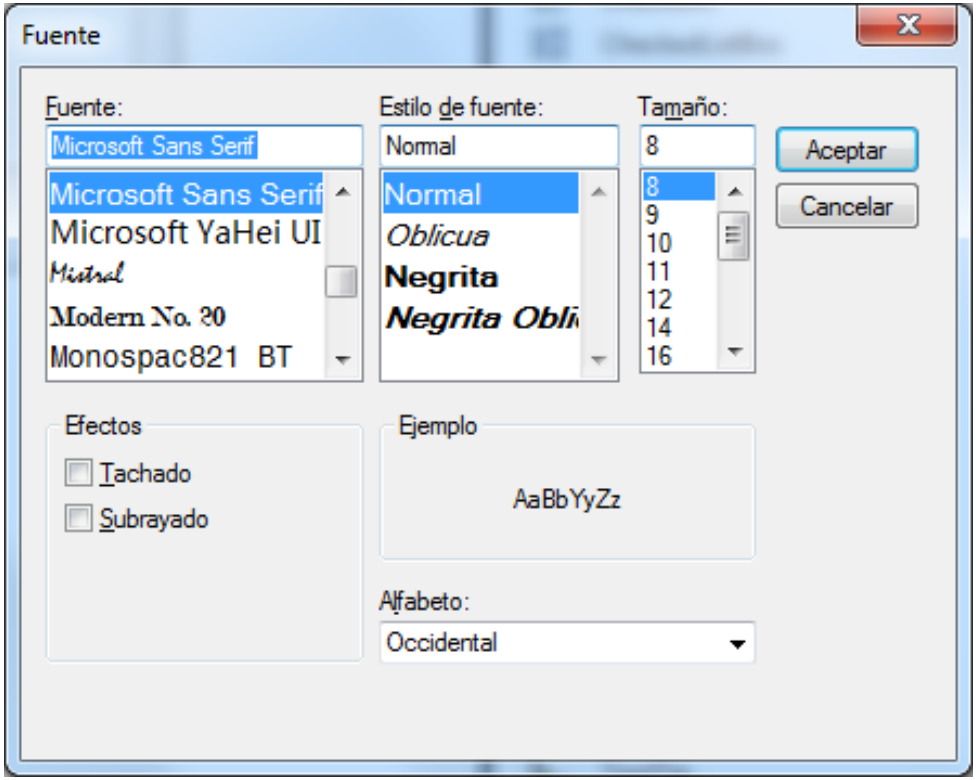
Asignación de la propiedad ForeColor

1. Click on ForeColor.
2. Click on button at right of settings box.
3. Click on Custom tab to obtain display shown.
4. Click on a color.



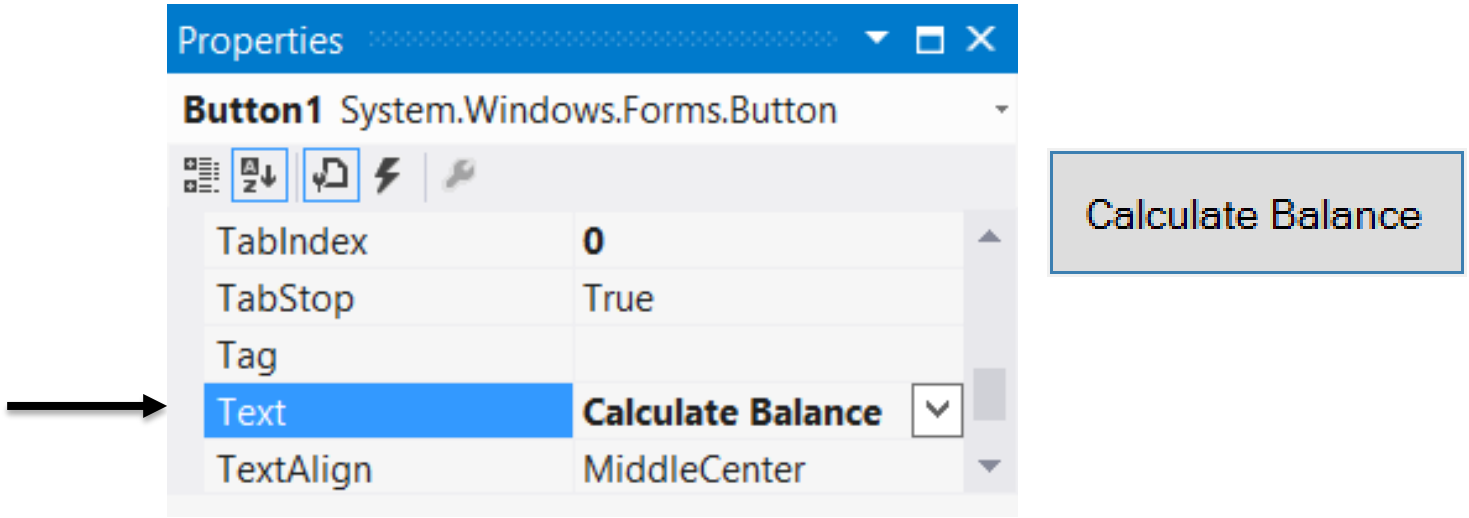
Asignación de la propiedad ForeColor

1. Click on Font in left column.
2. Click on ellipsis at right of settings box to obtain display shown.
3. Make selections and click on *OK*..

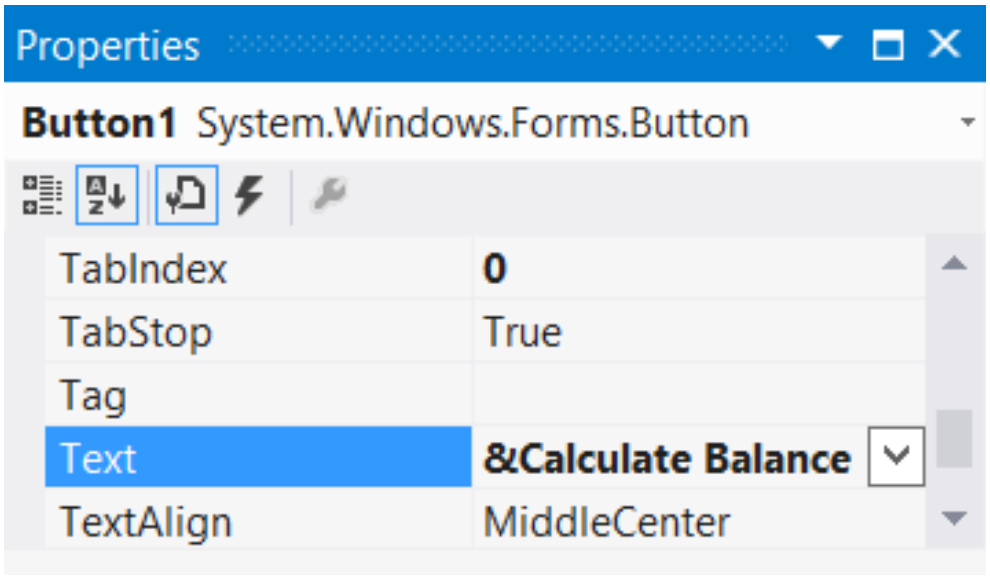


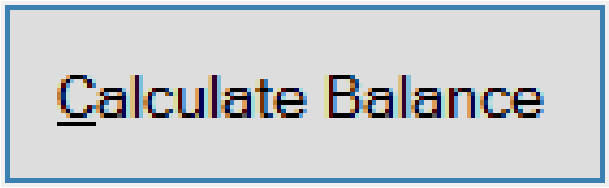
Control Button

- The caption on the button should indicate the effect of clicking on the button.



Añadiendo tecla de acceso





Control Label

- Used to identify the contents of a text box.
- Text property specifies caption.
- By default, label automatically resizes to accommodate caption on one line.
- When the AutoSize property is set to False, label can be resized manually. AutoSize is used primarily to obtain a multi-rowed label.

Control List Box

- Initially used to display several pieces of output.
- Also, it is used to select from a list.

Propiedad Name

- Used by the programmer to refer to a control in code
- Setting for Name property near top of Properties window
- Use appropriate 3-character naming prefix
- Use descriptive names

Prefijos del Control Name

Control	Prefix	Example
button	btn	btnCompute
label	lbl	lblAddress
text box	txt	txtAddress
list box	lst	lstOutput

Renombrando el Form

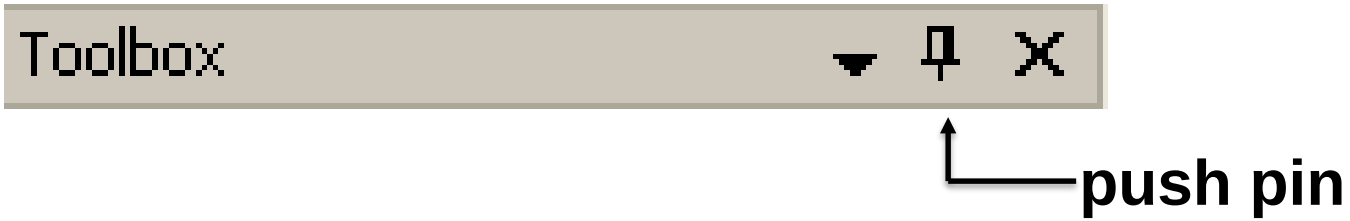
- Initial name is Form1
- The Solution Explorer window lists a file named Form1.vb.
- To rename the form, change the name of this file to *newName*.vb
- *newName* should begin with prefix *frm*.

Fonts

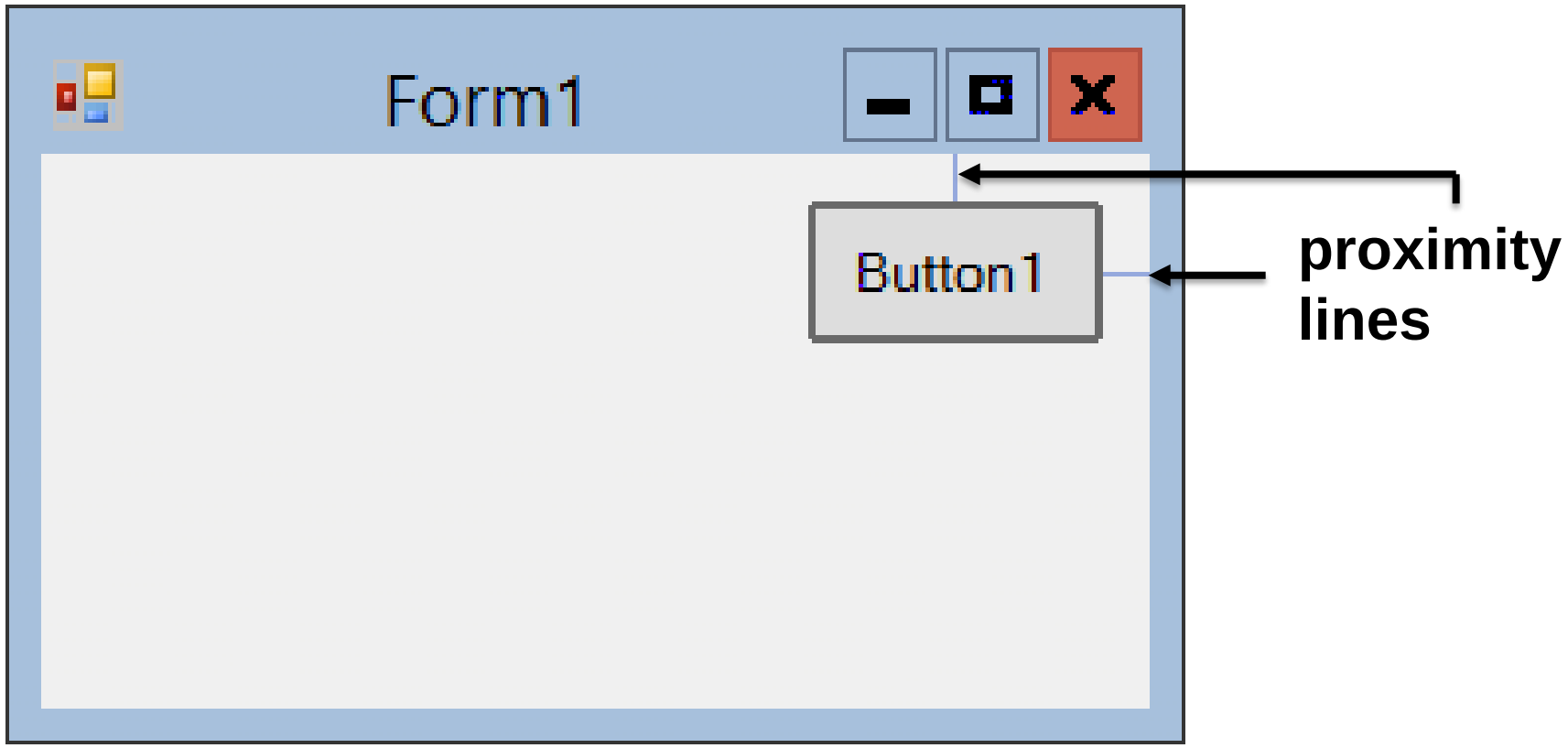
- Proportional width fonts, such as Microsoft Sans Serif, use less space for "I" than for "W“.
- Fixed-width fonts, such as Courier New, take up the same amount of space for each character.
- Fixed-width fonts are used for tables.

Auto Hide

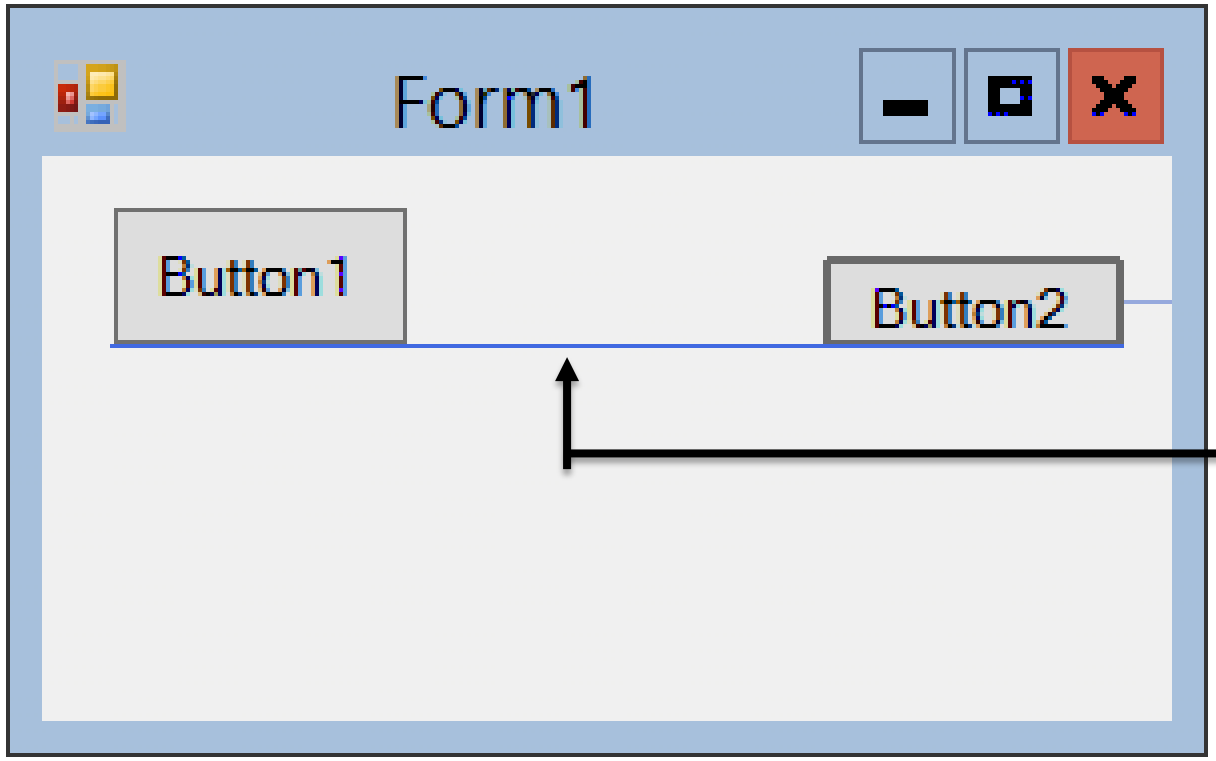
- Hides Toolbox when not in use
- Vertical push pin icon indicates auto hide is disabled.
- Click the push pin to make it horizontal and enable auto hide.



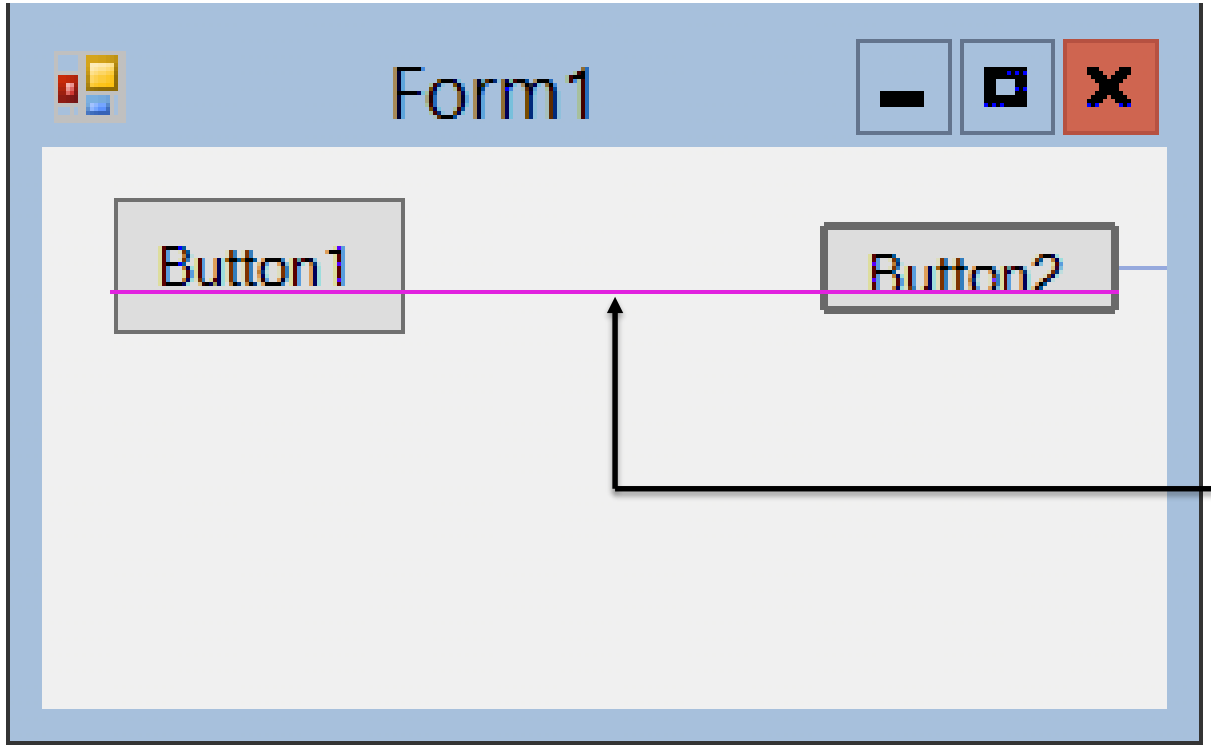
Posicionamiento de Controles



Alineamiento de Controles

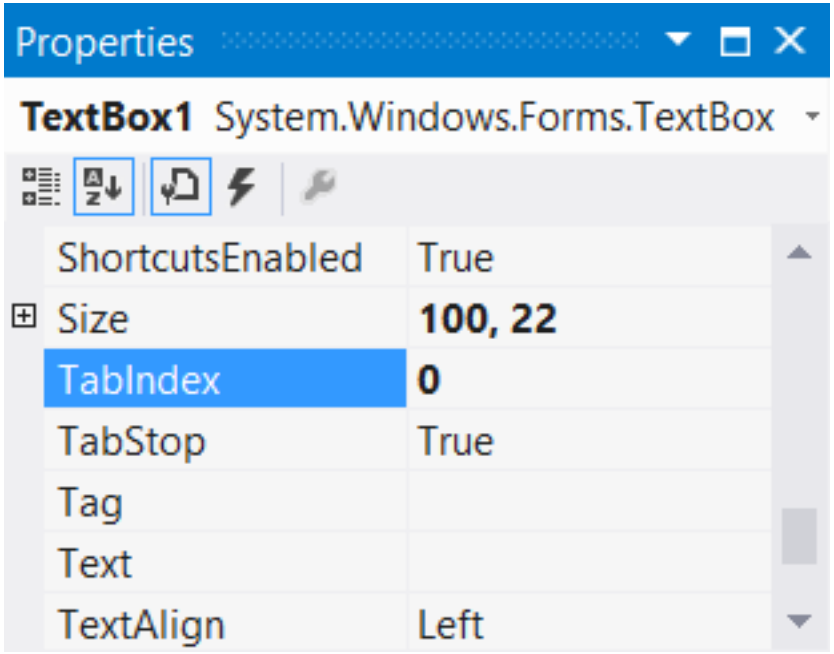


Alineamiento de Controles



Tab Order

- The tab indices determine the order in which controls receive the focus during tabbing.
- The control whose TabIndex property is set to 0 has the focus when the program begins.



Evento

- An **event** is an action, such as the user clicking on a button
- Usually, nothing happens in a Visual Basic program until the user does something and raises an event.
- What happens is determined by statements inside the event procedure.

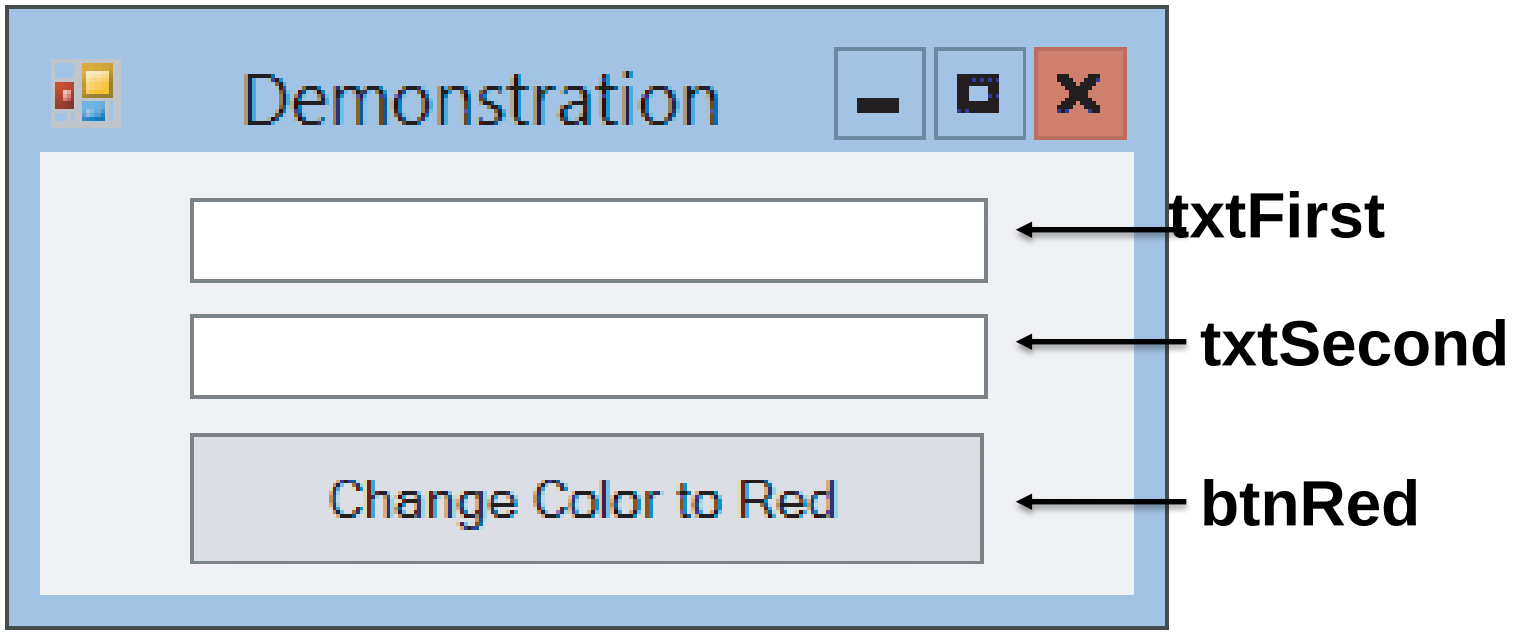
Ejemplos de Instrucciones

- txtBox.ForeColor = Color.Red
- txtBox.Visible = True
- txtBox.Text = "Hello World"

General Form:

controlName.property = setting

Ejemplo de Form



Focus

- When you click on a text box, a cursor appears in the text box, and you can type into the text box.
- Such a text box is said to have the **focus**.
- If you click on another text box, the first text box loses the focus and the second text box receives the focus.

Ejemplos de eventos

- btnShow.Click
- txtBox.TextChanged
- txtBox.Leave

General Form:

controlName.event

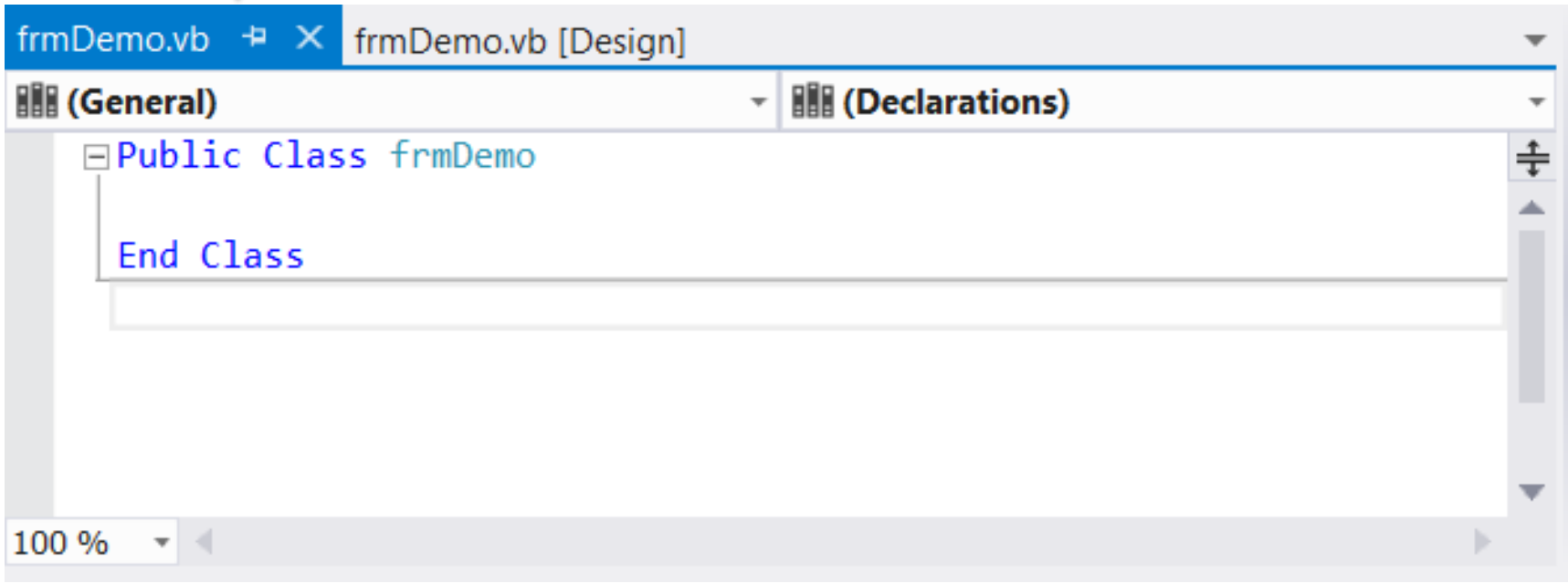
Pasos para crear un programa Visual Basic

1. Create the interface; that is, generate, position, and size the objects.
2. Set properties; that is, configure the appearance of the objects.
3. Write the code that executes when events occur.

Editor de código

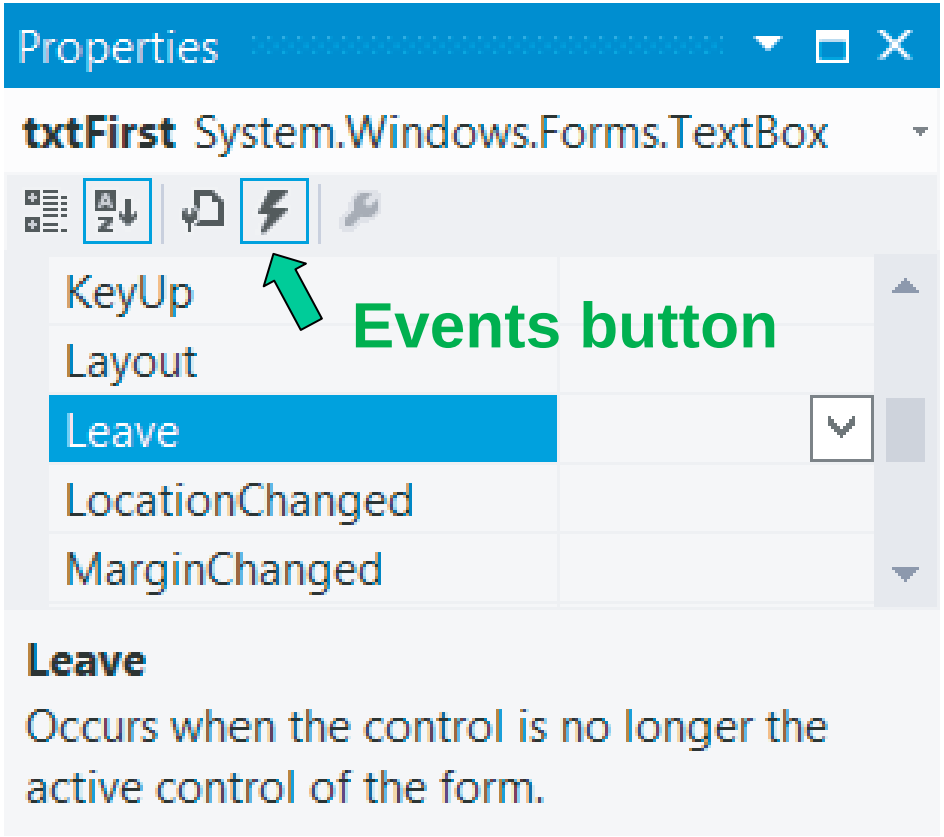
Code Editor tab

Form Designer tab



Mostrando eventos para un control

- Select the control
- Click on the Events button in the Properties window



Estructura de un Event Procedure

header
```
Private Sub objectName_event(...)
                    Handles objectName.event
    statements
End Sub
```

`(...)` is filled automatically with `(sender As System.Object, e As System.EventArgs)`

Creación del esquema para un Event Procedure

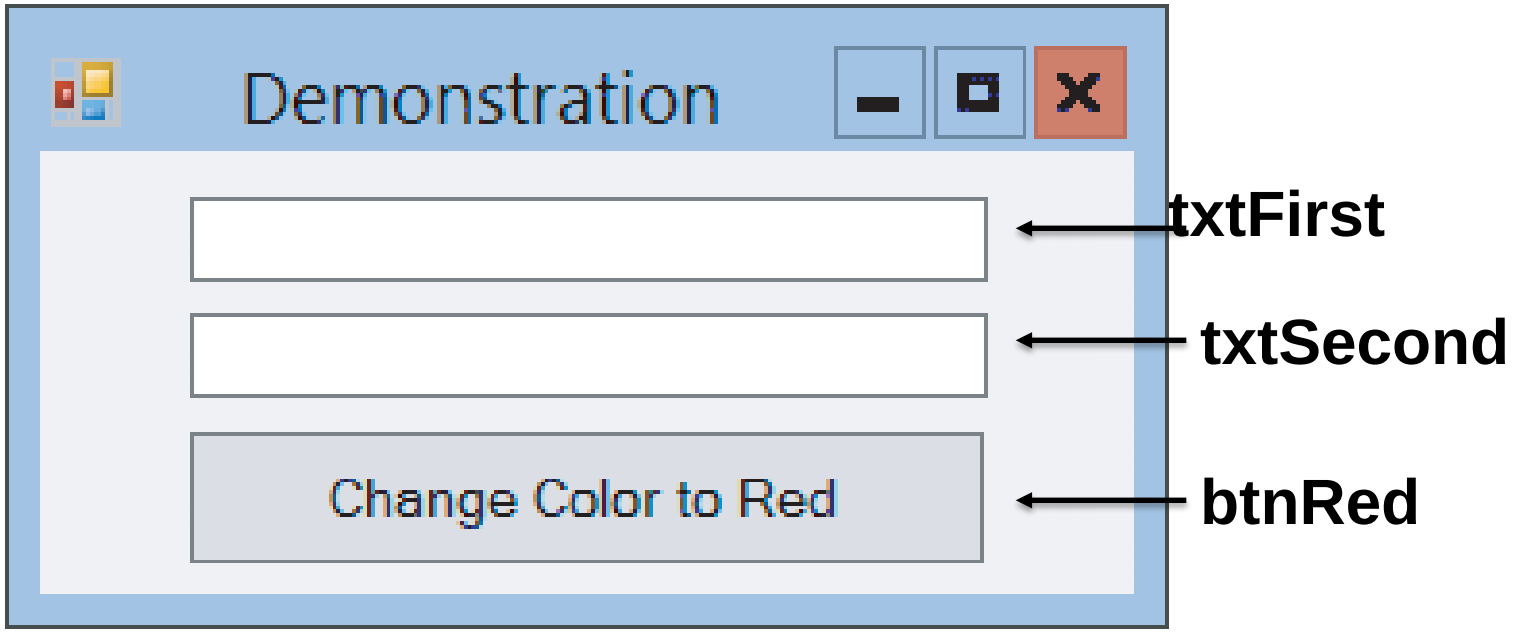
- Double-click on a control

  or

- Select a control, click on the Events button in the Properties window, and double-click on an event

(We nearly always use the first method.)

Ejemplo de Form



Double-click on txtFirst to create the outline for the Code Editor

Código

```
Public Class frmDemo
  Private Sub txtFirst_TextChanged(...)
              Handles txtFirst.TextChanged
    txtFirst.ForeColor = Color.Blue
  End Sub
End Class
```

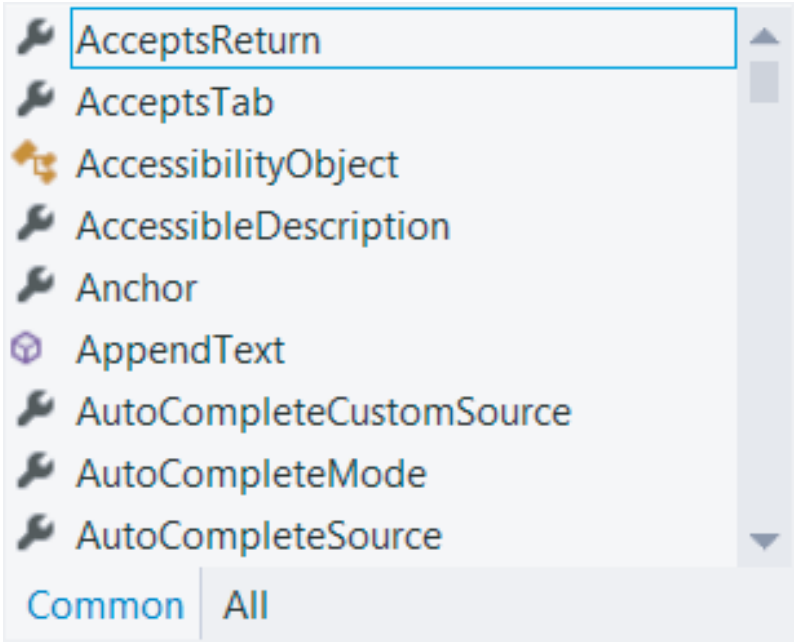
Comentarios

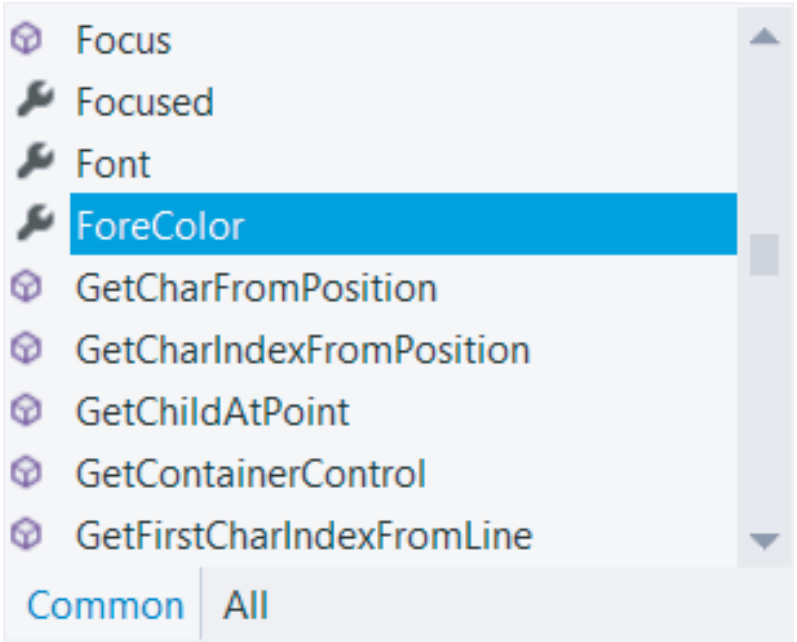
- *Comments* or *remarks* are short notes that you can write in the application's code to explain what the code does
- A comment starts with an apostrophe (**'**)
- Anything appearing after the apostrophe, to the end of the line, is ignored by the compiler. A comment can also be inserted at the end of a programming statement

IntelliSense

- Automatically pops up to help the programmer.

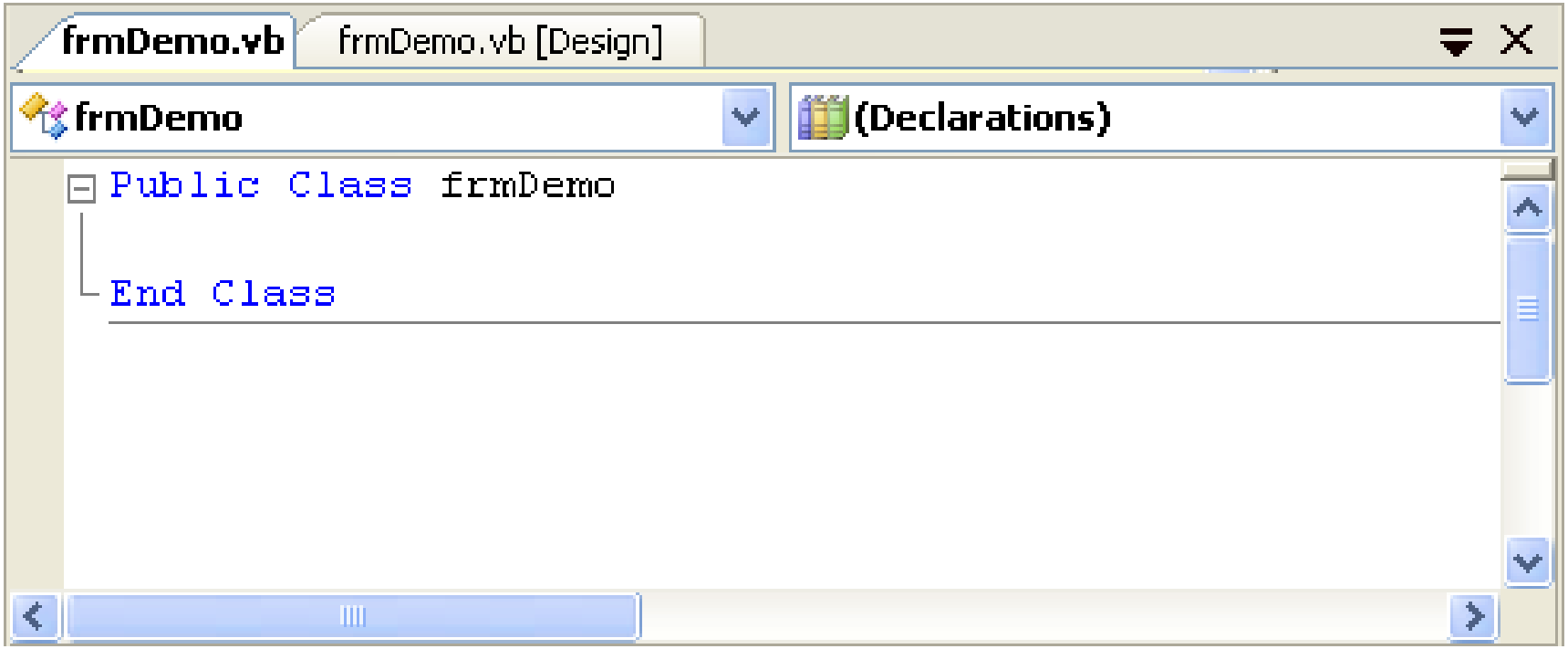
```
txtFirst.
```



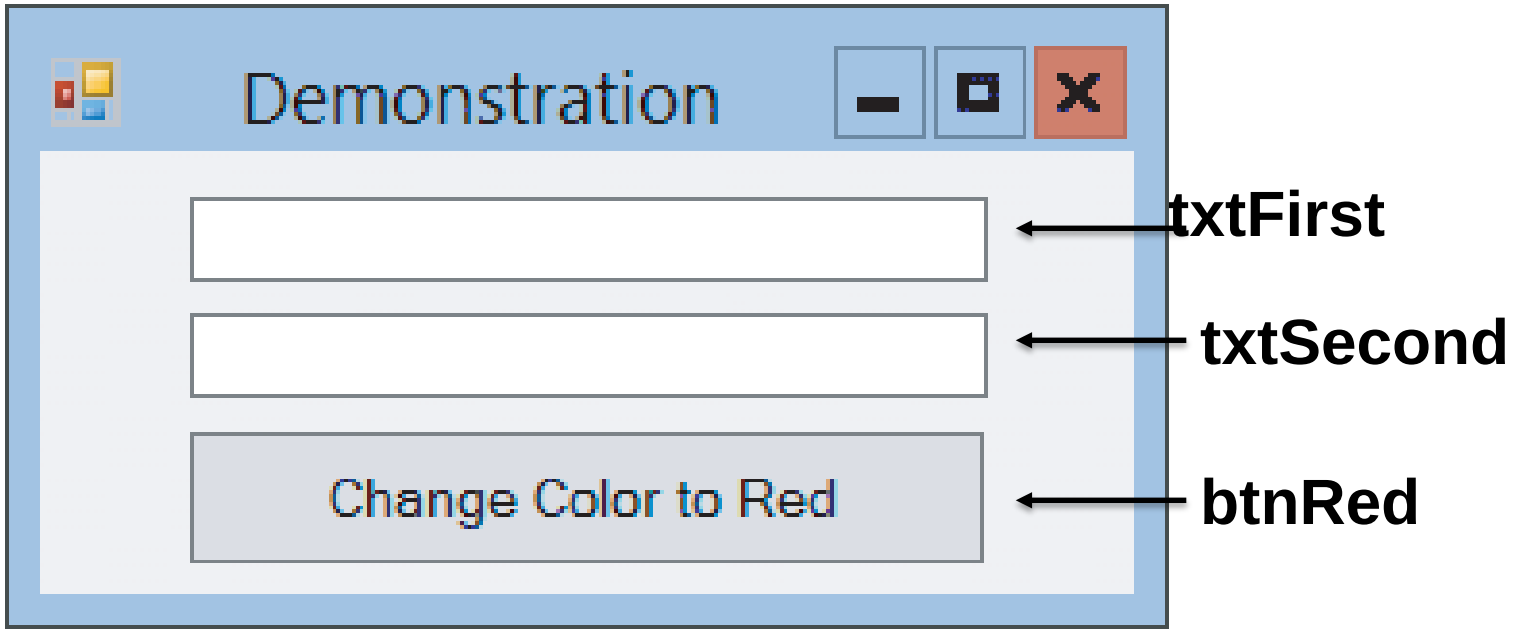


Editor de Código

- Click this tab to return to Form Designer.



Ejemplo de Form



Double-click on btnRed to return to Code Editor and add the outline of an event procedure.

Código

```
Public Class frmDemo
  Private Sub txtFirst_TextChanged(...)
              Handles txtFirst.TextChanged
    txtFirst.ForeColor = Color.Blue
  End Sub

  Private Sub btnRed_Click(...)
              Handles btnRed.Click
    txtFirst.ForeColor = Color.Red
  End Sub
End Class
```

Event Procedure txtFirst.Leave

- Select txtFirst on the form
- Click on the Events button in the Properties window
- Double-click on Leave

Código

```
Private Sub txtFirst_Leave(...)
                                Handles txtFirst.Leave
  txtFirst.ForeColor = Color.Black
End Sub

Private Sub txtFirst_TextChanged(...)
                        Handles txtFirst.TextChanged
  txtFirst.ForeColor = Color.Blue
End Sub

Private Sub btnRed_Click(...) Handles btnRed.Click
  txtFirst.ForeColor = Color.Red
End Sub
```

Cabecero de un Event Procedure

```
Private Sub btnRed_Click(…) Handles btnRed.Click
```

Name, can be changed.

Identifies event

```
Private Sub Button_Press(…) Handles btnRed.Click
```

Gestión de eventos múltiples

An event procedure can be invoked by two events.

```
Private Sub Happening(...) _
        Handles btnRed.Click,txtSecond.Leave
  txtFirst.ForeColor = Color.Red
End Sub
```

Alterando las propiedades del Form

- The following won't work:

```
frmDemo.Text = "Demonstration"
```

- The form is referred to by the keyword *Me*.

```
Me.Text = "Demonstration"
```

Abrir y Ejecutar un programa existente

- Click on *Open Project* in the *File* menu.
- Navigate to the program's folder.
- Double-click on the program's folder to open it.
- Double-click on the file with extension *sln*.
- In the Solution Explorer double-click on the file with extension *vb*. (The Form Designer will appear.)
- Press F5 to run the program.

Variables, Input y Output

Operaciones aritméticas – Expresiones numéricas

- Numbers are called **numeric literals**
- Five arithmetic operations in Visual Basic

  \+ addition

  \- subtraction

  * multiplication

  / division

  ^ exponentiation

- Expresiones numéricas:

  2 + 3

  3 * (4 + 5)

  2 ^ 3

Mostrando números

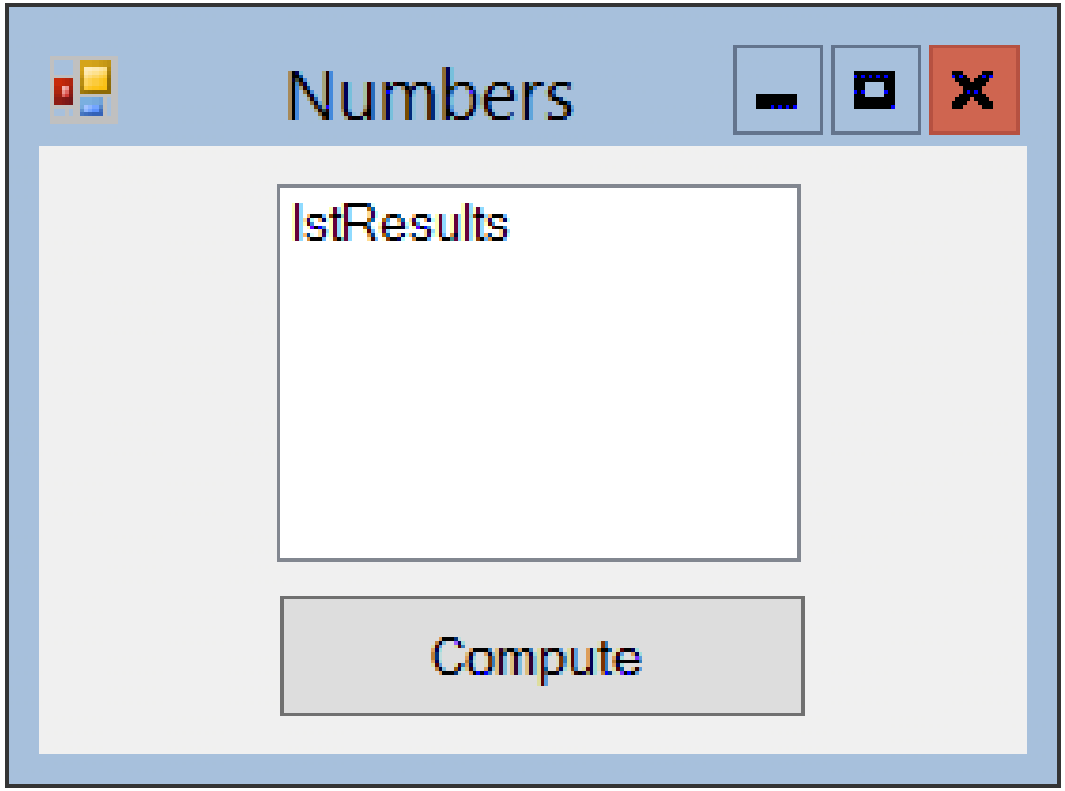
Let *n* be a number or a numeric expression.

The statement

```
lstBox.Items.Add(n)
```

displays the value of *n* in the list box.

Ejemplo 1



Ejemplo 1: código y salida

```
Private Sub btnCompute_Click (...)
                          Handles btnCompute.Click
   lstResults.Items.Add(5)
   lstResults.Items.Add(2 * 3)
   lstResults.Items.Add((2 ^ 3) – 1)
End Sub
```

Output in list box

5
6
7

Variables numéricas - declaración

- A **numeric variable** is a name to which a number can be assigned.
- Examples:

  speed distance interestRate balance

- Declaration:

```
Dim speed As Double
```

variable name **data type**

- Assignment:

```
speed = 50
```

Reglas y convenciones para nombrar Variables

- The first character of a variable name must be a letter or an underscore. Subsequent characters may be a letter, underscore, or digit
  - Thus variable names cannot contain spaces or periods (or many other kinds of characters)
- Visual Basic keywords cannot be used as variable names
- The 1st letter of each subsequent word in the variable name should be capitalized
  - `intHoursWorked` - an integer variable
  - `strLastName` - a String variable

Tipos de Datos en Visual Basic

- Integer types
  - `Byte`
  - `Short`
  - `Integer`
  - `Long`
- Floating-Point types
  - `Single`
  - `Double`
  - `Decimal`
- Other data types
  - `Boolean`
  - `Char`
  - `String`
  - `Date`

Tipo de dato Integer

- For values that will always be a whole number
- Usually name a variable starting with a 3 or 4 letter prefix indicating the variable's type

Data Type	Naming Prefix	Description
Byte	`byt`	Unsigned integer from 0 to 255
Short	`shrt`	Signed integer from -32,768 to 32,767
Integer	`int`	Signed integer from -2,147,483,648 to 2,147,483,647
Long	`lng`	Signed integer from -9,223,372,036,854,775,808 to 9,223,372,036,854,775,807

Tipo de dato Floating-Point

- For values that may have fractional parts
- Single used most frequently
- Double used in scientific calculations
- Decimal often used in financial calculations

Data Type	Naming Prefix	Description
Single	`sng`	As large as 10^{38} plus or minus, 7 decimal positions
Double	`dbl`	As large as 10^{308} plus or minus,15 decimal positions
Decimal	`dec`	As large as 10^{29} plus or minus, 29 decimal positions

Otros tipos de datos

- Boolean – variable naming prefix is `bln`
  - Holds 2 possible values, *True* or *False*
- Char – variable naming prefix is `chr`
  - Holds a single character
  - Allows for characters from other languages
- String – variable naming prefix is `str`
  - Holds a sequence of up to 2 billion characters
- Date – variable naming prefix is `dat` or `dtm`
  - Can hold date and/or time information

Tipo de dato String

- A string literal is enclosed in quotation marks
  - The following code assigns the name Jose Gonzales to the variable `strName`

```
Dim strName as String
strName = "Jose Gonzales"
```

- An empty string literal can be coded as:
  - Two consecutive quotation marks

```
strName = ""
```

  - Or by the special identifier `String.Empty`

```
strName = String.Empty
```

Tipo de dato Date

- Date data type variables can hold the date and time or both
  - You can assign a date literal to a Date variable, as shown here:

```
Dim dtmBirth As Date
dtmBirth = #5/1/2010#
```

- A date literal is enclosed within # symbols
  - All of the following Date literals are valid:

```
#12/10/2010#
#8:45:00 PM#
#10/20/2010 6:30:00 AM#
```

Variables numéricas – inicialización y uso

- Numeric variables are automatically initialized to 0:

```
Dim varName As Double
```

- To specify a nonzero initial value

```
Dim varName As Double = 50
```

Numeric variables can be used in numeric expressions.

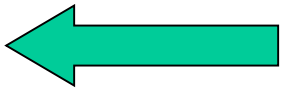
```
Dim balance As Double = 1000

lstBox.Items.Add(1.05 * balance)
```

Output: **1050**

Variables numéricas – asignación e incremento

```
Dim numVar1 As Double = 5
Dim numVar2 As Double = 4
numVar1 = 3 * numVar2
lstBox.Items.Add(numVar1)
```



Output: **12**

- To add 1 to the numeric variable *var*

  ```
  var = var + 1
  ```

- Or as a shortcut

  ```
  var += 1
  ```

  Or as a generalization

  ```
  var += numeric expression
  ```

Operadores aritméticos

Visual Basic operation	Arithmetic operator	Algebraic expression	Visual Basic expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm	`b * m`
Division (floating point)	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Division (integer)	\	none	`v \ u`
Modulus	Mod	$r \bmod s$	`r Mod s`
Exponentiation	^	q^p	`q ^ p`
Unary minus	-	$-e$	`-e`
Unary plus	+	$+g$	`+g`

Operadores de asignación combinados

- These special assignment operators provide an easy means to perform these common operations:

Operator	Usage	Equivalent to	Effect
+=	x += 2	x = x + 2	*Add to*
-=	x -= 5	x = x – 5	*Subtract from*
*=	x *= 10	x = x * 10	*Multiply by*
/=	x /= y	x = x / y	*Divide by*
\=	x \= y	x = x \ y	*Int Divide by*
&=	a &= b	a = a & b	*Concatenate*

Operadores aritméticos - precedencia

Operator(s)	Operation	Order of evaluation (precedence)
^	Exponentiation	Evaluated first. If there are several such operators, they're evaluated from left to right.
+, -	Sign operations (unary)	Evaluated second. If there are several such operators, they're evaluated from left to right.
*, /	Multiplication and Division	Evaluated third. If there are several such operators, they're evaluated from left to right.
\	Integer division	Evaluated fourth. If there are several such operators, they're evaluated from left to right.
Mod	Modulus	Evaluated fifth. If there are several such operators, they're evaluated from left to right.
+, -	Addition and Subtraction (binary)	Evaluated sixth. If there are several such operators, they're evaluated from left to right.

Agrupando con paréntesis

- Parentheses ( ) can be used to force selected parts of an expression to be evaluated before others
  - Assume we're computing the average of 3 numbers
  - `dblAvg = int1 + int2 + int3 / 3 ' incorrect`
  - `int3 / 3` is evaluated first
  - That result is added to `int1` and `int2`
- Use parentheses to control order of operations
  - `dblAvg = (int1 + int2 + int3) / 3 ' correct`
  - `int1 + int2 + int3` is evaluated first
  - That result is divided by 3
- When in doubt, use parentheses!

Operadores comparación y relacionales

Standard algebraic equality operator or relational operator	Visual Basic equality or relational operator	Example of Visual Basic condition	Meaning of Visual Basic condition
Equality operators			
=	=	x = y	x is equal to y
≠	<>	x <> y	x is not equal to y
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y

Operadores lógicos

- Logical operators can be used to combine multiple Boolean expressions into a compound expression

Operator	Effect
`And`	Combines two expressions into one. Both expressions must be true for the overall expression to be true.
`Or`	Combines two expressions into one. One or both expressions must be true for the overall expression to be true. It is only necessary for one to be true, and it does not matter which.
`Xor`	Combines two expressions into one. One expression (not both) must be true for the overall expression to be true. If both expressions are true, or both expressions are false, the overall expression is false.
`Not`	Reverses the logical value of an expression: makes a true expression false and a false expression true.

Operadores - precedencia

Operators	Type
^	exponentiation
+ -	sign operations (unary)
* /	multiplication and floating-point division
\	Integer division
Mod	modulus
+ -	addition and subtraction (binary)
= <> < <= > >=	equality and relational

Literales

Type	Description	Example
Boolean Keywords	*True* and *False*	`True`
Byte	Decimal digits between *0* and *255*	`200`
Char	Character surrounded by double quotes followed by lowercase *C*	`"A"c`
Date	Date and/or time representation enclosed in #	`#1/1/14#`
Decimal	Digits with decimal point followed by *D* or @	`+32.0D`
Double	Digits with decimal point followed by optional *R*	`3.5R`
Integer	Decimal digits followed by optional letter *I*	`-3054I`
Long	Decimal digits followed by the letter *L*	`40000L`
Short	Decimal digits followed by the letter *S*	`12345S`
Single	Digits with decimal point followed by letter *F* or *!*	`26.4F`
String	Characters surrounded by double quotes	`"ABC123"`

Conversiones de tipo implícitas

- A value of one data type can be assigned to a variable of a different type
  - An implicit type conversion is an attempt to convert to the receiving variable's data type
- A widening conversion suffers no loss of data
  - Converting an integer to a double
  - `Dim dblVal As Double = 5`
- A narrowing conversion may lose data
  - Converting a decimal to an integer
  - `Dim intNum As Integer = 12.2 ' intNum becomes 12`

Constantes declaradas

- Can declare a variable whose value is set at declaration and cannot be changed later:

```
Const dblSALES_TAX_RATE As Double = 1.06
```

- Looks like a normal declaration except:
  - `Const` used instead of `Dim`
  - An initialization value is required
  - By convention, entire name capitalized with underscore characters to separate words
- The objective of our code is now clearer

```
Const dblSALES_TAX_RATE As Double = 1.06
dblTotal *= dblSALES_TAX_RATE
```

Conversiones de tipo explícitas

- The following narrowing conversions require an explicit type conversion
  - Double to Single
  - Single to Integer
  - Long to Integer
- Boolean, Date, Object, String, and numeric types represent different sorts of values and require conversion functions as well

Conversiones de tipo explícitas - ejemplos

- Rounding can be done with the `CInt` function

```
intCount = CInt(12.4) ' intCount value is 12
intCount = CInt(12.5) ' intCount value is 13
```

- `CStr` converts an integer value to a String

```
Dim strText As String = CStr(26)
```

- `CDec` converts a String to a Double

```
Dim dblPay As Double = CDbl("$1,500")
```

- `CDate` converts a String to a Date

```
Dim datHired As Date = CDate("9/14/2014")
```

Funciones para conversión de tipo

- Funciones de conversión comunes:

Function	Description
`Cint` (*expression*)	Converts *expression* to an Integer
`Cdbl` (*expression*)	Converts *expression* to a Double
`Cdate` (*expression*)	Converts *expression* to a Date
`Cdec` (*expression*)	Converts *expression* to a Decimal
`Cstr` (*expression*)	Converts *expression* to a String

Lista de funciones de conversión

- Hay funciones de conversión para cada tipo de dato:

`CBool` (*expression*)
`CByte` (*expression*)
`CChar` (*expression*)
`CDate` (*expression*)
`CDbl` (*expression*)
`CDec` (*expression*)

`CInt` (*expression*)
`CLng` (*expression*)
`CObj` (*expression*)
`CShort` (*expression*)
`CSng` (*expression*)
`CStr` (*expression*)

Instrucción de selección If

- Formato general:

```
If expression Then
   statement
   (more statements may follow)
End If
```

- If the expression is *True*, execute the statements between `If...Then` and `End If`
- Otherwise, the statements are skipped

Instrucción de selección If

- Ejemplo:

```
If decSales > 50000 Then
   MessageBox.Show("You've earned a bonus!")
   decCommissionRate = 0.12
   intDaysOff = intDaysOff + 1
End If
```

Instrucción de selección If - Else

- Formato general:

```
If expression Then
   statement
   (more statements may follow)
Else
   statement
   (more statements may follow)
End If
```

- If the expression is *True*
  - Execute the statements between `If...Then` and `Else`
- If the expression is *False*
  - Execute the statements between `Else` and `End If`

Instrucción de selección If – ElseIf múltiples

- Formato general:

```
If expression Then
  statement
  (more statements may follow)
ElseIf expression Then
  statement
  (more statements may follow)
(put as many ElseIf statements as necessary)
Else
  statement
  (more statements may follow)
End If
```

- This construction is like a chain of `If...Then...Else` statements. The `Else` part of one statement is linked to the `If` part of another

Instrucción de selección If – ElseIf ejemplo

```
' Display the letter grade.
If dblAverage < 60 Then
  lblGrade.Text = "F"
ElseIf dblAverage < 70 Then
  lblGrade.Text = "D"
ElseIf dblAverage < 80 Then
  lblGrade.Text = "C"
ElseIf dblAverage < 90 Then
  lblGrade.Text = "B"
ElseIf dblAverage <= 100 Then
  lblGrade.Text = "A"
Else
  lblGrade.Text = "Invalid Score"
End If
```

- A sequence of `ElseIf` statements may end with a plain `Else`, called a *trailing* `Else`
- If none of the conditions are *True*, the trailing `Else` statement(s) will be executed
- The trailing `Else` catches any value that falls through the cracks

Instrucción Select Case

- Similar to `If…Then…ElseIf`
  - Performs a series of tests
  - Conditionally executes the first *true* condition
- `Select Case` is different in that:
  - A single test expression may be evaluated
  - The test expression is listed once
  - The possible values of the expression are then listed with their conditional statements
- `Case Else` may be included and executed if none of the values match the expression

Instrucción Select Case

- Formato general:

```
Select Case TestExpression
  [Case ExpressionList
    [one or more statements]]
  [Case ExpressionList
    [one or more statements]]
  ' Case statements may be repeated
  ' as many times as necessary.
  [Case Else
    [one or more statements]]
End Select
```

Instrucción Select Case ejemplo

```
Select Case CInt(txtInput.Text)
  Case 1
    MessageBox.Show("Day 1 is Monday.")
  Case 2
    MessageBox.Show("Day 2 is Tuesday.")
  Case 3
    MessageBox.Show("Day 3 is Wednesday.")
  Case 4
    MessageBox.Show("Day 4 is Thursday.")
  Case 5
    MessageBox.Show("Day 5 is Friday.")
  Case 6
    MessageBox.Show("Day 6 is Saturday.")
  Case 7
    MessageBox.Show("Day 7 is Sunday.")
  Case Else
    MessageBox.Show("That value is invalid.")
End Select
```

Instrucción Select Case ejemplo expresiones múltiples y operadores relac.

```
Select Case intNumber
  Case 1, 3, 5, 7, 9
    strStatus = "Odd"
  Case 2, 4, 6, 8, 10
    strStatus = "Even"
  Case Else
    strStatus = "Out of Range"
End Select
```

```
Select Case dblTemperature
  Case Is <= 75
    blnTooCold = True
  Case Is >= 100
    blnTooHot = True
  Case Else
    blnJustRight = True
End Select
```

Instrucción Select Case ejemplo rango de valores

```
Select Case intScore
  Case Is >= 90
    strGrade = "A"
  Case 80 To 89
    strGrade = "B"
  Case 70 To 79
    strGrade = "C"
  Case 60 To 69
    strGrade = "D"
  Case 0 To 59
    strGrade = "F"
  Case Else
    MessageBox.Show("Invalid Score")
End Select
```

Instrucciones de repetición

- A *repetition structure*, or loop causes one or more statements to repeat
- Each repetition of the loop is called an *iteration*
- Visual Basic has three types of loops:
  - `Do While`
  - `Do Until`
  - `For… Next`
- The difference among them is how they control the repetition

Instrucción Do While

- Formato

```
Do While condition
   statement(s)
Loop
```

Condition is tested,
If it is true,
the loop is run.
If it is false,
the statements
following the
Loop statement
are executed.

These statements are inside the body of the loop and are run if the condition above is true.

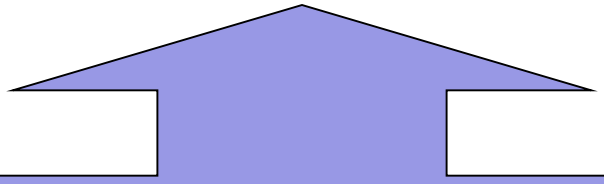
Instrucción Do While ejemplo

```
Private Sub btnDisplay_Click(...) _
                        Handles btnDisplay.Click
  'Display the numbers from 1 to 7
  Dim num As Integer = 1
  Do While num <= 7
    lstNumbers.Items.Add(num)
    num += 1  'Add 1 to the value of num
  Loop
End Sub
```

Do Until

- Formato

```
Do
   statement(s)
Loop Until condition
```



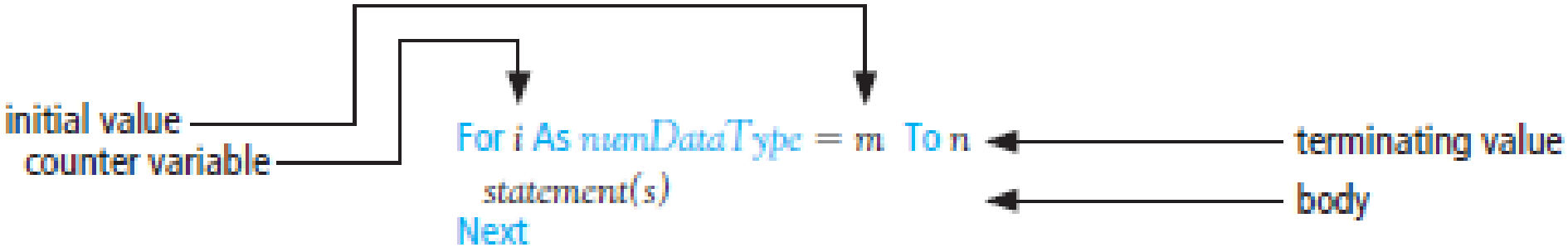
Loop is executed once and then the condition is tested. If it is false, the loop is run again. If it is true, the statements following the Loop Until statement are executed.

Instrucción Do Until ejemplo

```
Dim passWord As String = ""
Do
   passWord = InputBox("What is the password?")
   passWord = passWord.ToUpper
Loop Until passWord = "SHAZAM"
```

Ciclo For…Next

- Used when we know how many times we want the loop to execute
- A counter controlled loop



Ciclo For…Next

```
For i As Integer = 1 To 5
   lstTable.Items.Add(i & " " & i ^ 2)
Next
```

The loop counter variable, i, is

- initialized to 1
- tested against the stop value, 5
- incremented by 1 at the Next statement

```
Dim i As Integer = 1
Do While i <= 5
   lstTable.Items.Add(i & " " & i ^ 2)
   i += 1
Loop
```

Ciclo For…Next Step keyword

- Normally after each pass the value of the counter variable increases by 1
- If **`Step s`** is appended to the For statement, the value of **`s`** will be added to the counter variable after each pass.
- If the value of **`s`** is a negative number, the value of the counter variable will decrease after each pass.

Ciclo For…Next Step keyword

```
For j As Integer = 10 To 1 Step -1
   lstBox.Items.Add(j)
Next
lstBox.Items.Add("Blastoff")
```

Salida forzada de un ciclo

- In some cases it is convenient to end a loop before the test condition would end it
- The following statements accomplish this
  - `Exit Do`
    - (used in `Do While` or `Do Until` loops)
  - `Exit For`
    - (used in `For`…`Next` loops)
- Use this capability with caution
  - It bypasses normal loop termination
  - Makes code more difficult to debug

¿Qué ciclo usar?

- Each type of loop works best in different situations
  - The `Do While` loop
    - When you wish the loop to repeat as long as the test expression is true or at least once as a pretest loop
  - The `Do Until` loop
    - When you wish the loop to repeat as long as the test expression is false or at least once as a pretest loop
  - The `For…Next` loop
    - Primarily used when the number of required iterations is known

Modularidad en VB

Visual Basic has two devices for breaking problems into smaller pieces:

- **Function** procedures: is a collection of statements that performs a task and returns a value to the part of the program that executed it. Example: Visual Basic's built-in functions, such as `CInt` and `IsNumeric`
- **Sub** procedures: is a collection of statements that performs a task. Example: A *procedure* Event handlers are a type of procedure
- A *method* can be either a procedure or a function.

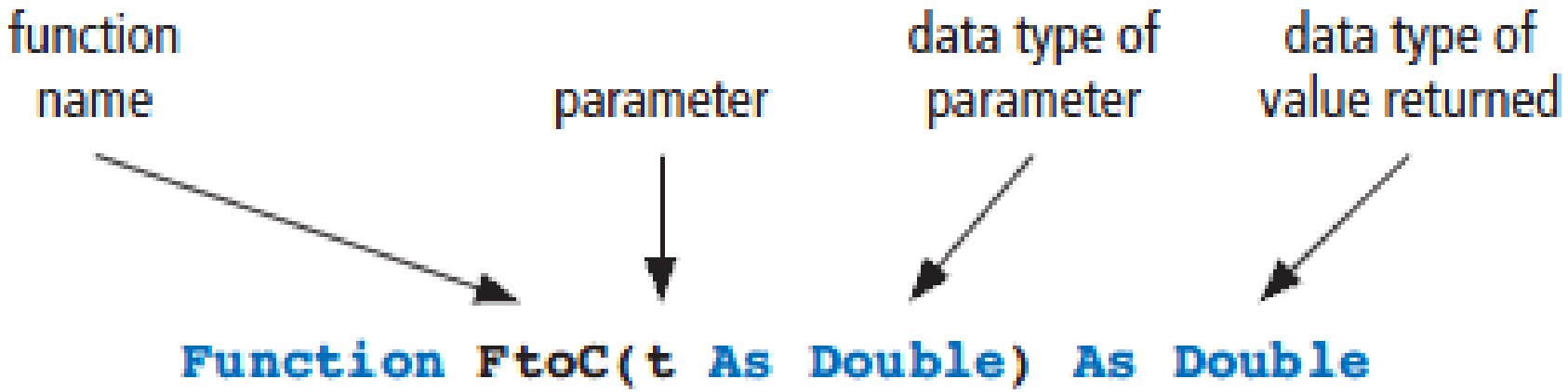
Funciones

- Function procedures (aka user-defined functions) always return one value
- Syntax:

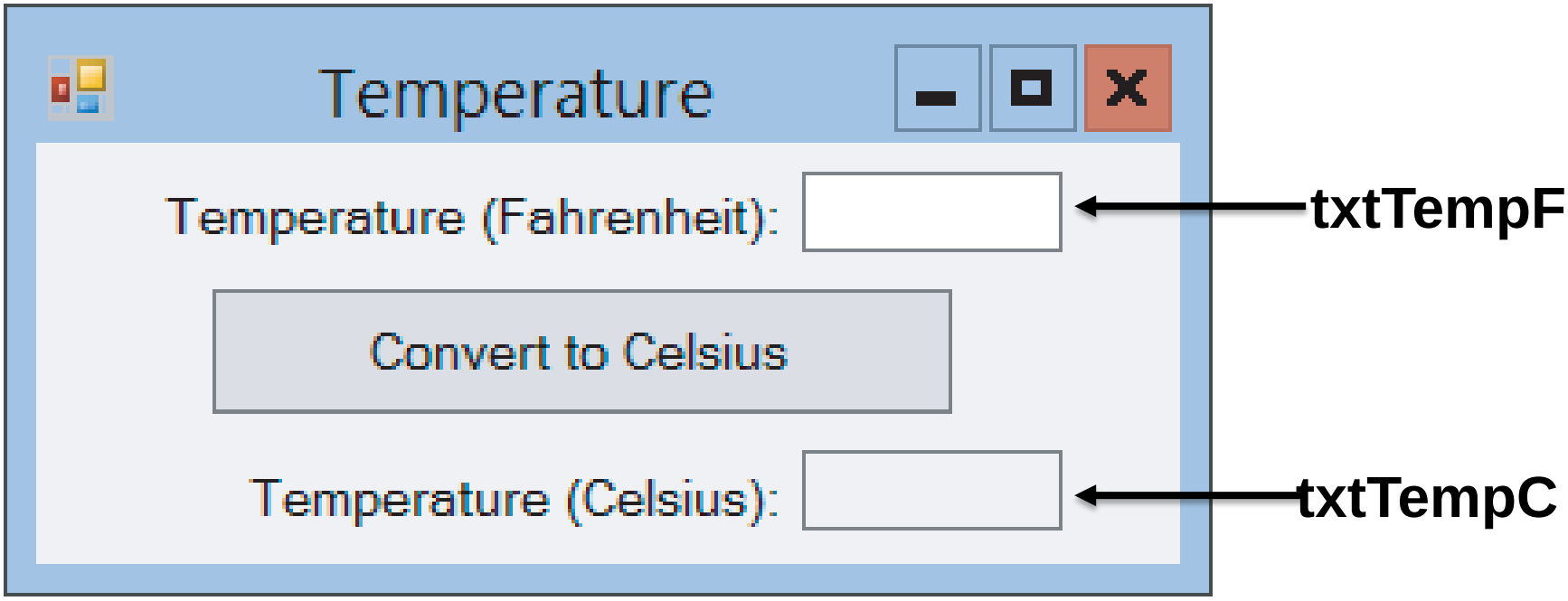
```
Function FunctionName(var1 As Type1,
                      var2 As Type2,
                      ...) As
  ReturnDataType
   statement(s)
   Return expression
End Function
```

Funciones – ejemplo 1 parámetro

```
Function FtoC(t As Double) As Double
  'Convert Fahrenheit temp to Celsius
  Return (5 / 9) * (t - 32)
End Function
```



Funciones – ejemplo 1 parámetro



Funciones – ejemplo 1 parámetro

```
Private Sub btnConvert_Click(...) _
                        Handles btnConvert.Click
  Dim fahrenheitTemp, celsiusTemp As Double
  fahrenheitTemp = CDbl(txtTempF.Text)
  celsiusTemp = FtoC(fahrenheitTemp)
  txtTempC.Text = CStr(celsiusTemp )
End Sub

Function FtoC(t As Double) As Double
  Return (5 / 9) * (t - 32)
End Function
```

Procedimientos Sub

- Formato general:

```
Sub ProcedureName(par1 As Type1, par2 As Type2, _
                                        ..., parN As TypeN)
    statement(s)
End Sub
```

- The statement that invokes a Sub procedure is referred to as a calling statement.
- A calling statement looks like this:

```
ProcedureName(arg1, arg2,..., argN)
```

Paso de parámetros: por valor

- When a variable argument is passed to a parameter **not** preceded with ByRef, just the value of the argument is passed.
- After the Sub procedure terminates, the variable has its original value.

Paso de parámetros: por valor - ejemplo

```
Public Sub btnOne_Click (...) Handles _
                                    btnOne.Click
  Dim n As Double = 4
  Triple(n)
  txtBox.Text = CStr(n)
End Sub

Sub Triple(num As Double)
  num = 3 * num
End Sub
```

Output: **4**

Paso de parámetros: por referencia

- When a variable argument is passed to a parameter preceded by ByRef, the parameter is given the same memory location as the argument.
- After the Sub procedure terminates, the variable has the value of the parameter. Temporary copy of the original argument

Paso de parámetros: por valor - ejemplo

```
Public Sub btnOne_Click (...) Handles _
                                    btnOne.Click
  Dim n As Double = 4
  Triple(n)
  txtBox.Text = CStr(n)
End Sub

Sub Triple(ByRef num As Double)
  num = 3 * num
End Sub
```

Output: **12**

Arrays

- An **array variable** is a collection of simple variables of the same type to which VB can efficiently assign a list of values.
- An array stores multiple values of same type
- For example, the days of the week might be:
  - A set of 7 string variables with a maximum length of 9 characters.
- All variables within an array are called *elements* and must be of the same data type
- You access the elements in an array through a *subscript*

Características de indexación

- A *subscript*, also called an *index*, is a number that identifies a specific element within an array
- Subscript numbering works like a list box index:
  - Subscript numbering begins at 0
  - 1st element in an array is always subscript 0
  - Last element is total *number of elements – 1*
- An array with 7 elements refers to the 1st element as subscript 0 and the last element as subscript 6

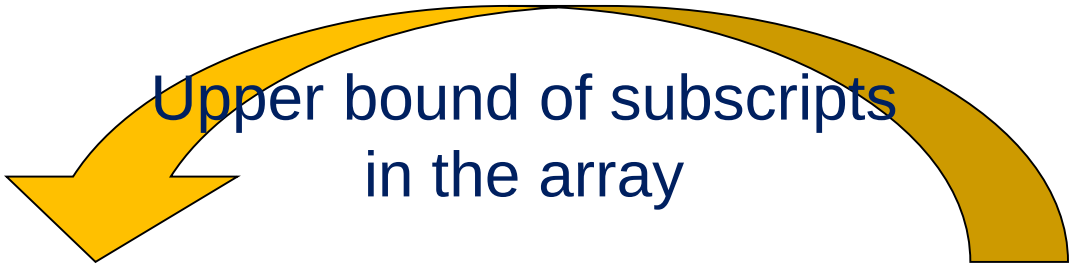
Declaración de Array

- Declare an array much like a regular variable

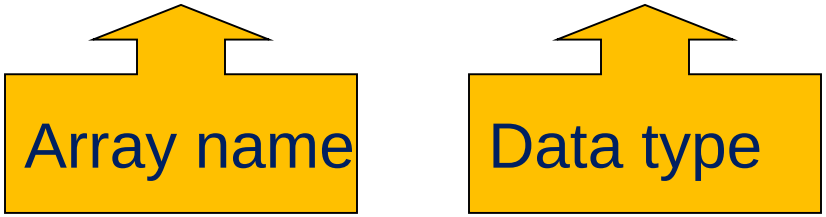
```
Dim ArrayName (UpperSubscript) As DataType
```

  - *ArrayName* is the name of the array
  - *UpperSubscript* is the value of the array's highest subscript
    - Must be a positive Integer
    - Positive Integer named constant
    - Integer variable containing a positive number
  - *DataType* is a Visual Basic data type

Usando Arrays




```
Dim students(29) As String
Dim scores(29) As Double
```



Inicialización de arrays

Arrays may be initialized when created:

```
Dim arrayName() As DataType =
            {value0, value1, value2, ..., valueN}
```

declares an array having upper bound *N* and assigns *value0* to *arrayName*(0), *value1* to *arrayName*(1), ..., and *valueN* to *arrayName*(*N*).

Example:

```
Dim teamNames() As String =
        {"Packers", "Packers", "Jets", "Chiefs"}
```

Métodos de arrays

arrayName.Count	number of elements
arrayName.Max	highest value
arrayName.Min	lowest value
arrayName.First	first element
arrayName.Last	last element

- The upper bound of *arrayName* is **`arrayName.Count - 1`**
- **`arrayName.First`** is the same as **`arrayName(0)`**

Métodos de arrays numéricos

numArrayName.Average	average value of elements
numArrayName.Sum	sum of values of elements

Uso de ciclos en lugar de métodos

```
Dim ages() As Integer = {55, 56, 61, 52,
  69,64, 46, 54, 47}    'last 9 presidents
Dim max As Integer = ages(0)
For i As Integer = 1 To ages.Count - 1
   If ages(i) > max Then
      max = ages(i)
   End If
Next
txtOutput.Text = "Greatest age: " & max
```

Output: **Greatest age: 69**

Array bidimensional

- One-dimensional arrays store a list of items of the same type
- Two-dimensional arrays store a table of items of the same type.
- Consider the rows of the table as numbered 0, 1, 2, ,,, *m* and the columns numbered 0, 1, 2, …, *n*. Then the array is declared with

```
Dim arrayName(m, n) As DataType
```

Item in *i*th row, *j*th column: **`arrayName(i,j)`**

Array bidimensional - ejemplo

```
Dim rm(,) As Double = {{0, 2054, 802, 738},
                       {2054, 0, 2786, 2706},
                       {802, 2786, 0, 100},
                       {738, 2706, 100, 0}}
```

declares and initializes an array of road-mileages. Some elements of the array are

rm(0,0)=0, rm(0,1)=2054, rm(1,2)=2786

Lectura de datos desde ficheros texto

- Data stored in a text file can be read one line at a time with a StreamReader object.
- The following statement declares a variable of type StreamReader and specifies the file to be read from.

```
Dim srVar As IO.StreamReader =
                        IO.File.OpenText(filespec)
```

Note: A pointer is set to the first line of the file.

Lectura de datos desde ficheros texto

- **`strVar = srVar.ReadLine`** reads the line pointed to, assigns the line to the string variable *strVar*, and moves the pointer to the next line of the file.
- The value of **`srVar.EndOfStream`** will be True after the entire file has been read.
- The statement **`srVar.Close()`** terminates communication with the file.

Lectura de datos desde ficheros texto

If *srVar* is a variable of type StreamReader, an entire text file can be read with a loop of the form

```
Do Until srVar.EndOfStream
   strVar = srVar.ReadLine
   .
   .
Loop
```

Escritura en ficheros texto

- Data can be placed in a text file one line at a time with a StreamWriter object.
- The following statement declares a variable of type StreamWriter and specifies the file to be created.

```
Dim swVar As IO.StreamWriter =
                    IO.File.CreateText(filespec)
```

Escritura en ficheros texto

- `swVar.WriteLine(info)` initally places the information into the first line of the file.
- Subsequent statements of that form place information into lines at the end of the file.
- The statement `swVar.Close()` terminates communication with the file.

Adición de elementos en ficheros texto

1. Execute the statement

```
Dim swVar As IO.StreamWriter = _
                    IO.File.AppendText(filespec)
```

   where *filespec* identifies the file.

2. Add lines of data to the end of the file with the WriteLine method.
3. After all the data have been written into the file, close the file with **`swVar.Close()`**.

Note: If the file does not exist, the AppendText method will create it.

Modos de ficheros texto

- OpenText – open for input
- CreateText – open for output
- AppendText – open for append
- A file should not be opened in two different modes at the same time.

Comprobación de la existencia de un fichero

```
Dim sr As IO.StreamReader
If IO.File.Exists(filespec) Then
  sr = IO.File.OpenText(filespec)
Else
  message = "Either no file has yet been "
  message &= "created or the file named"
  message &= filespec & " is not found."
  MessageBox.Show(message, "File Not Found")
End If
```

Métodos de borrado y renombrado

- Delete method:

```
IO.File.Delete(filespec)
```

- Move method (to change the filespec of a file):

```
IO.File.Move(oldfilespec, newfilespec)
```

- Note: The IO.File.Delete and IO.File.Move methods cannot be used with open files.

Imports System.IO

- Simplifies programs that have extensive file handling.
- Place the statement **`Imports System.IO`**

  at the top of the Code Editor, before the **`Class frmName`** statement. Then, there is no need to insert the prefix "IO." before the words StreamReader, StreamWriter, and File.

Medir tiempo de ejecución de código

- A Stopwatch instance can measure elapsed time for one interval, or the total of elapsed time across multiple intervals.
- In a typical Stopwatch scenario, you call the Start method, then eventually call the Stop method, and then you check elapsed time using the Elapsed property.

Medir tiempo de ejecución de código Ejemplo

```
Dim time As Stopwatch = Stopwatch.StartNew
time.Start()
' Measure.
For i As Integer = 0 To 1000 - 1
   Threading.Thread.Sleep(1)
Next
' Stop measuring.
time.Stop()
L_out.Text = "Time: " &
  time.ElapsedMilliseconds.ToString & " msec"
time.Reset()
time.Start()
…
time.Stop()
```

Referencias

- An Introduction to Programming Using Visual Basic 2012. Ninth Edition by David I. Schneider
- Starting out with Visual Basic 2012. Sixth Edition by Tony Gaddis, Kip Irvine