



Objectives:

- ❖ Get in touch with the **Visual Basic (VB) working environment** (Spanish version)
- ❖ Graphical design of a VB program: **command button (cmd)**
 - Properties: **Name** (Nombre) and **Caption**,
 - Events: **Click**
- ❖ Design of **flowcharts** and **coding sequential** programs
- ❖ Finishing a VB program: End
- ❖ Definition of **constants** and **variables**: Const and Dim
- ❖ Reading and writing instructions: **InputBox** and **MsgBox**
- ❖ String, Integer and Double data types and conversions CInt and Cdbl
- ❖ **Assignment** and simple **expressions**
- ❖ String **concatenation** and newline: **&** and vbCrLf

Greeting program

Our first exercise consists in **designing and implementing** a Visual Basic (VB) program that asks for the name and answers with a greeting. We will provide a solution for this problem.

Interface

Figure 1.1 shows the proposed interface for our program, which will consist of two types of command button controls labeled “Greet me” and “Quit”, to carry out the respective actions.

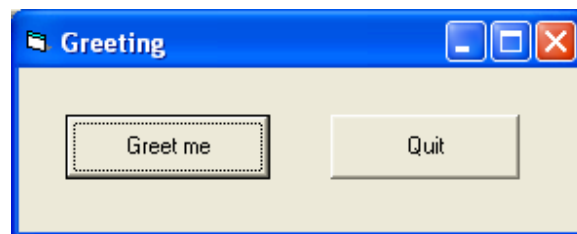


Figure 1.1 Greeting program interface.

Flowcharts

For the purposes of the program we are going to identify the buttons as `cmdGreet` and `cmdQuit`. We form the name of the control adding before the acronym `cmd` (command) to facilitate the identifying task when we have many controls.

Each of these buttons will have an associated subprogram. When we click on these buttons, the code associated with that event will be executed (the “click” event).

Figure 1.2 shows the flowchart of the button ending the program. It is the simplest flowchart we shall use throughout all our VB programs in similar situations. The “EndProgram” action is a VB command we will see later in the section on coding.

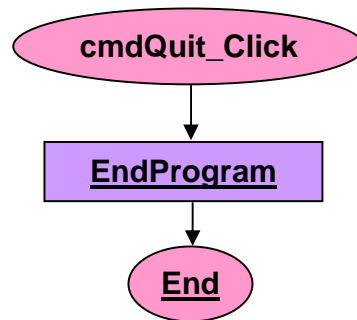


Figure 1.2 Flowchart associated with the `cmdQuit` button.

The flowchart of greeting button is shown in Figure 1.3. We declare two **string** variables called `name` and `greeting`. We read the name and calculate the greeting by concatenating the string "Hi ", the typed name and the exclamation mark "!". Note that we use the "+" symbol to express the concatenation of strings, and even though it is valid in VB, we use a different symbol, as we shall see later.

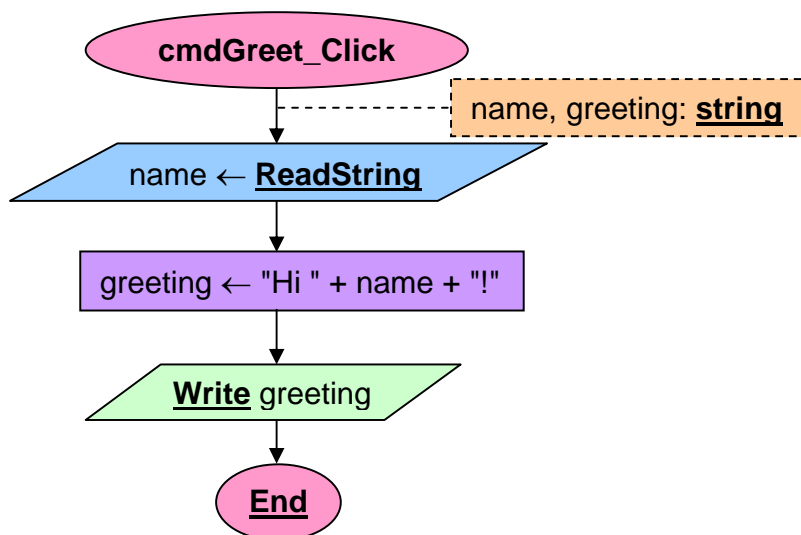


Figure 1.3 Flowchart of the `cmdGreet` button.

Codification

A sub-program begins with the **sub** instruction followed by the name of the subprogram, `cmdQuit_Click ()`; it ends with the order **End Sub**.

The code associated with the **Click** event on the `cmdQuit` button is simple, as shown in Figure 1.4. Simply call the VB **End** instruction, which corresponds to the order "**EndProgram**" that had been specified in the flowchart in Figure 1.2.

```

Sub cmdQuit_Click()
  End
End Sub
  
```

Figure 1.4 Code of the `cmdQuit` button.

The code associated with the **Click** event on the cmdGreet button in Figure 1.5 corresponds to the flowchart in Figure 1.3. We can identify the different elements.

```

Sub cmdGreet_Click()
  Dim name As String, greet As String
  name = InputBox ("Introduce your name")
  greet = "Hi " & name & "! " 'Construct greeting
  MsgBox greet
End Sub

```

Figure 1.5 Code of the cmdGreet button.

First declare the variable name and greet as a character strings (**String**).

Then we request the user to enter his/her name, using the **InputBox** VB instruction, collecting the result in the variable name. The outcome will be that we will display a dialog box like the Figure 1.6.

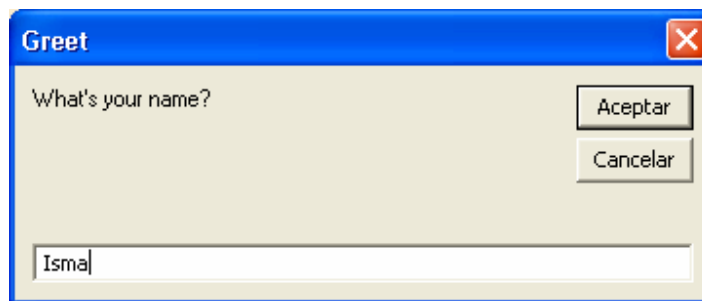


Figure 1.6 Dialogue box for the **InputBox** instruction.

We calculate the text to display by concatenating the literal "Hi " string and the entered name, "Isma" at the example. The concatenation operator in VB is the '&' *ampersand* sign.

After that we show in a dialog box (**MsgBox**) with a greeting as shown in Figure 1.7.

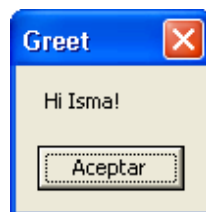


Figure 1.7 Dialogue box for the **MsgBox** instruction.

Entering the program in the Spanish VB environment

You can create your first program by following these steps:

1. Launch the program: Inicio → Todos los Programas → Microsoft Visual Basic 6.0 → Microsoft Visual Basic 6.
2. If you see a dialog box like Figure 1.8 to select "Exe estándar" → Open

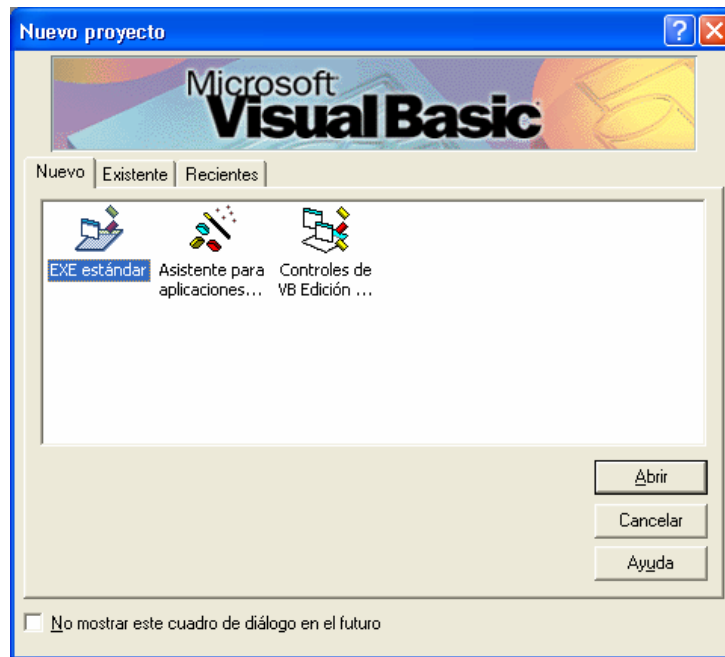


Figure 1.8 New Visual Basic project.

3. Menú → Ver → Cuadro de Herramientas (in the bottom part if it is not selected). The resulting environment is shown at figure 1.9

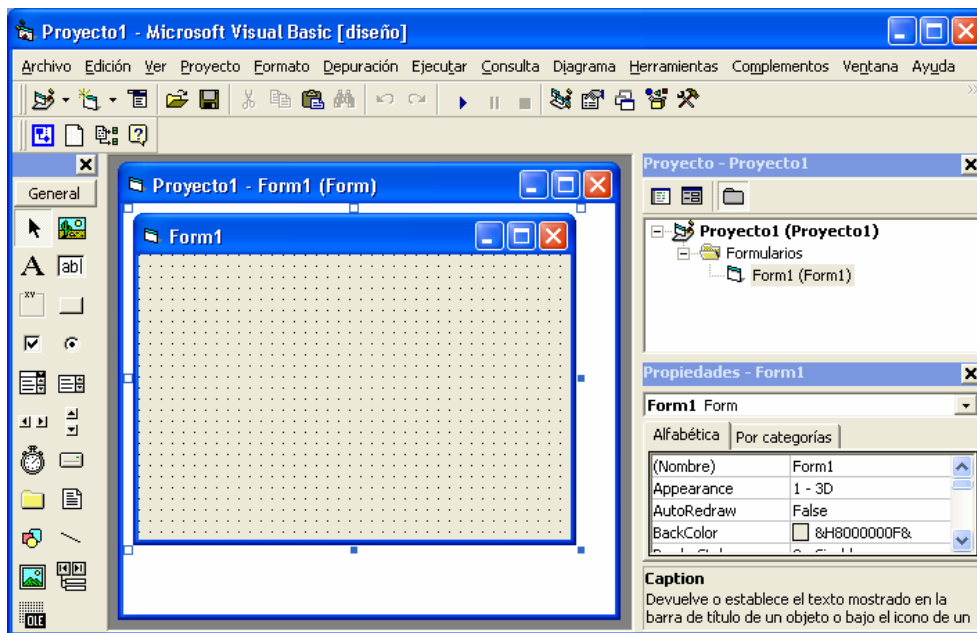



Figure 1.9 Visual Basic environment.

4. Select (click) Table Tools button  (Figure 1.10) and draw a button on the form, by clicking and dragging.

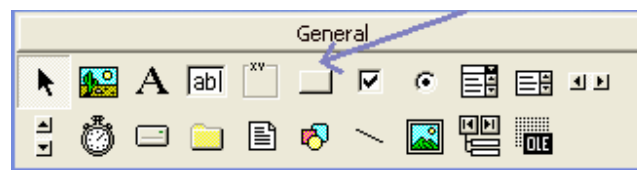


Figure 1.10 Button in toolbar.

5. In the panel properties appear properties button. We will modify the following:
 - a. **(Nombre):** name of the button in the code, `cmdGreet`
 - b. **Caption:** text in the button, `Salúdame`
6. We will do likewise with the Quit button:
 - a. **(Nombre):** `cmdQuit`
 - b. **Caption:** Quit
7. If we double-clicked `cmdGreet` there appears an edition of the skeleton code with the corresponding sub-program, as shown in Figure 1.11. It will introduce the code in Figure 1.5. Note that in the proposed code in the previous line says "Option Explicit" and also puts "Private" to the keyword "Sub". We will keep without going to discuss its usefulness

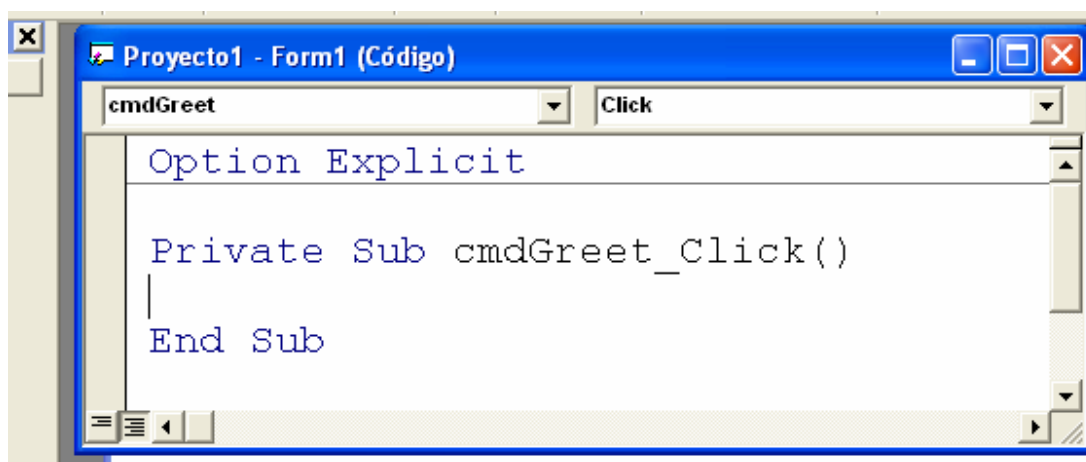



Figure 1.11 Code editing.

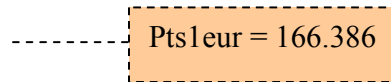
8. For execution: press F5, click  or pick in the menu (Ejecutar → Iniciar)
9. For saving every project it is recommended to use pen-drive with USB interface:

Menú → Archivo → Guardar Proyecto

 - i. Name of **form**: Greet (.frm)
 - ii. Name of the **project**: Greet (.vbp) – it is displayed in **InputBox** and **MsgBox**
 - iii. You can delete the unnecessary archives. (.vbw)
10. After that you only have to do this to save: Archivo → Guardar.

Constants

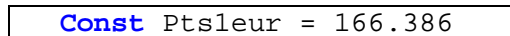
To declare a constant, for example, the relationship between the euro and the old pesetas, put, together with the variable declarations (if any), the name and the value assigned by an equal, as shown in Figure 1.12.



```
Pts1eur = 166.386
```

Figure 1.12 Declaration of a constant.

The code to write in Visual Basic is illustrated in Figure 1.13.



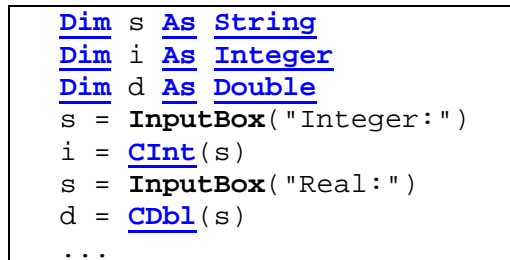
```
Const Pts1eur = 166.386
```

Figure 1.13 Initialisation code of a constant.

Reading integer and real numbers

The Visual Basic `InputBox` instruction returns a string. To read integer or real numbers we first read the text on a string variable and then we may convert them to the corresponding types with the `CInt` and `CDBl` respectively.

In Figure 1.14 we show an illustrative portion of code on how to read an integer `i` (in the flowchart **ReadInteger**) and a real `d` (in the flowchart **ReadReal**).



```
Dim s As String
Dim i As Integer
Dim d As Double
s = InputBox("Integer: ")
i = CInt(s)
s = InputBox("Real: ")
d = CDBl(s)
...
```

Figure 1.14 Reading code for integer and real numbers.

An aspect to take into account is that to convert strings into real numbers with `CDBl`, Visual Basic expects numbers in the local format, that is, if our local system is configured to Spanish conventions, instead of a decimal point (as we must use in the program) we must enter a decimal comma. Additionally, for integers (in Visual Basic 2 bytes, 16 bits) there will be an overflow after 32768, which will not be considered an error in our program so far.

Graphic Design Tips

A good practice to obtain that all the graphic elements of a form (eg, buttons) are of equal size from the start is to decide the shape and size of one of them, using as a reference the one with the longest text.

Given the first element it may be copied to the clipboard by **Menú** → **Edición** → **Copiar** or press Control-C and paste to duplicate it as many times as necessary by **Menú** → **Edición** → **Pegar** or by pressing Control-V.

Be very careful because you will get a dialog box like the one in Figure 1.15, asking if you want to create a button matrix with the same name but different index; you must choose the button labelled “No”.

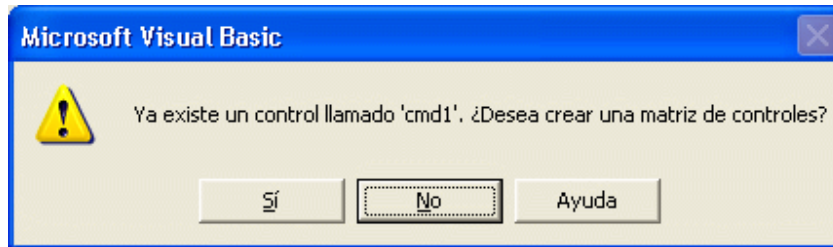


Figure 1.15 Dialog box after duplicating a button.

Proposed Exercises

For the following exercises use a single project, codifying each exercise in a different button (**cmd1**, **cmd2**, ..., **cmd9** y **cmdQuit**). In Figure 1.16 the interface is shown. You can check the behaviour with the provided executable.

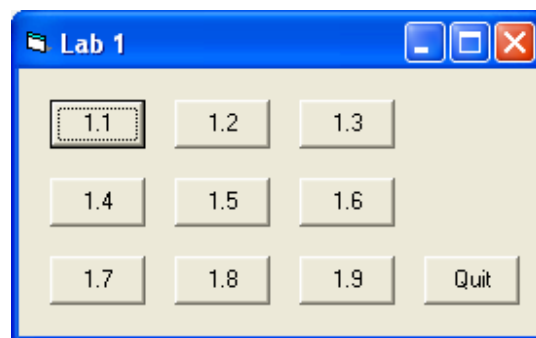


Figure 1.16 Interface of the proposed exercises.

1. **Design** the flowchart and **implement** a program to read two integer **numbers** and calculate their **sum**. Tests: 12+15, 12000 + 20000, 40000+12.
2. **Design** the flowchart and **implement** a program that reads the radius (r) and calculates the area (a) and the volume (v) of a sphere. Declare π (π) as a real constant (Double), worth 3.14159265358979. Display the result in a single dialog box in three separate lines (for radio, area and volume) for a newline (**vbCrLf**). Tests: $r = 1$, $r = 2$, $r = 45$, $r = 4.5$, $r = 4,5$.

$$a = 4\pi r^2 \quad v = 4\pi r^3/3$$

3. **Design** the flowchart and **implement** a program to read a integer number and displays the following five occurrences:

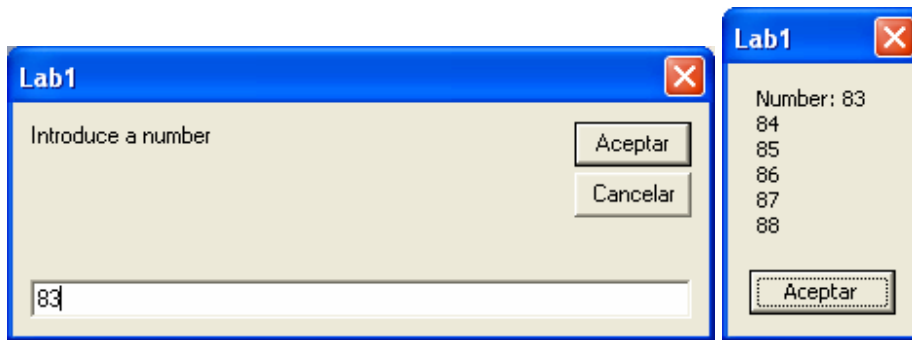


Figure 1.17 InputBox and MsgBox in exercise 3.

4. **Design and implement** a program to convert **Celsius** temperatures to **Fahrenheit**.

$$^{\circ}\text{C} = 5/9(^{\circ}\text{F}-32)$$
5. **Design and implement** a program to read **two integer numbers** and calculate their mean value.
6. **Design and implement** a program to read three integer numbers and calculate their mean value.
7. **Design and implement** a program to read **two integer numbers** and **exchange their values**, as shown below. Note that it is not enough to show the values in reverse order but change the contents of the variables in memory.

$$X = 23, Y = 45 \quad X = 45, Y = 23$$

8. **Design and implement** a program to read **three integer numbers** and **exchange their values**.

$$A = 23, B = 45, C = 67 \quad A = 45, B = 67, C = 23$$

9. **Design and implement** a program to read a number of seconds and show on the screen the corresponding **hours**, **minutes** and **seconds**. Example: If we introduce 4005 it will show "1 hour 6 minutes 45 seconds". If we enter a big number (> 32767) there will be an overflow.