Escuela Universitaria de Ingeniería Vitoria-Gasteiz
Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

Fundamentals of Computer Science
2010-2011
Laboratory 3
Conditional structures

erran ta zabal zazu

Objetives:
- **Design** of flowcharts for programs with **conditional sentences**
- **Implement** VB programs with **conditional sentences** (`If`, `If-Else`).
- Boolean expressions.
- Properties of the controls: `Visible`

Adder with check (1)

In the previous laboratory, in exercise 2.1 consisting in an adder, if we press straight on the "Add" button or if we introduce an arbitrary alphabetic string (see Figure 3.1) we obtain a VB error message like the one in Figure 3.2. In Figure 3.3 we show the code portion causing the error.

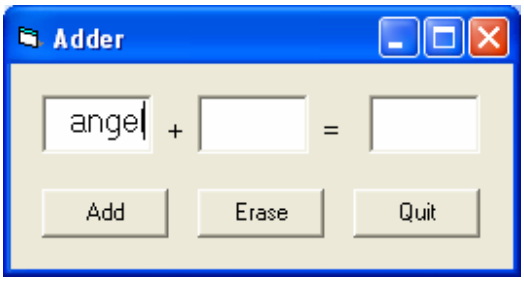

Figure 3.1 Incorrect data input in our adder.

Figure 3.2 Incorrect data type – runtime error (Spanish VB version)

```
Dim s As String
Dim op1 As Integer

s = txtOp1.Text
op1 = CInt (s)
```

Figure 3.2 Offending code portion.

The reason for this error (see Figure 3.3) is that in the assignment Visual Basic is trying to convert the contents of the `s` variable, a string, into an integer number (`op1`) and this is not possible under these circumstances, e.g. `CInt("angel")`.

Conditional instructions enable us to check if input data are adequate before continuing with calculations. To do this we can use the **Numeric** (*string*) predicate in flowcharts within the conditional clause. The corresponding flowchart can be seen in Figure 3.4.

Visual Basic provides a function called **`IsNumeric`** to carry out the proposed checking. This function receives a string and returns **`True`** if the string can be converted intro a numeric value and **`False`** otherwise.

The rest of the code will be as before. We provide the corresponding code in Figure 3.5.

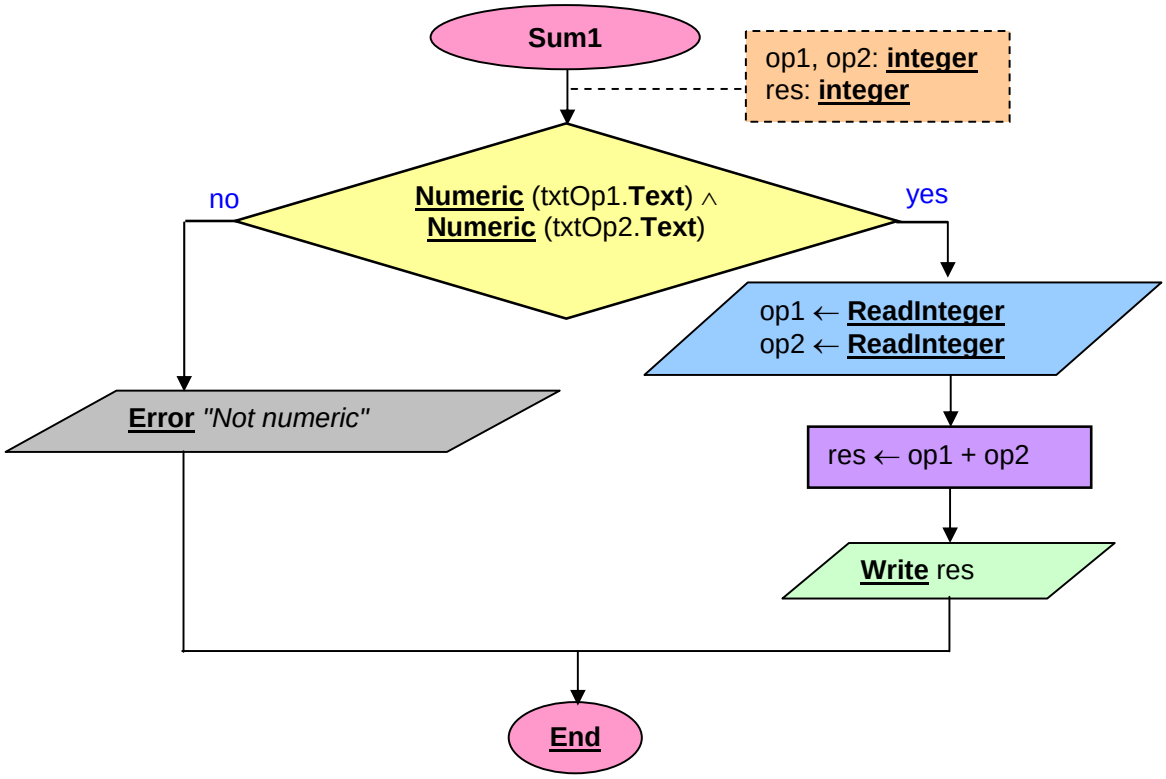

Figure 3.4 Flowchart of the sum with checking.

```
Sub cmdSum1_Click()
  Dim op1 As Integer, op2 As Integer
  Dim res As Integer

  If IsNumeric(txtOp1.Text) And _
     IsNumeric(txtOp2.Text) Then
    op1 = CInt (txtOp1.Text)
    op2 = CInt (txtOp2.Text)
    res = op1 + op2
    txtRes.Text = CStr (res)
  Else
    MsgBox "Operands must be numeric"
  End If
End Sub
```

Figure 3.5 VB code for the sum with checking.

Adder with check (2)

The way we have solved the problem in the previous section of our adder with check, the user receives no information on which of the operands is incorrect.

To provide this information we need two different checks with two separate error messages. The flowchart contemplating this possibility can be seen in Figure 3.6.

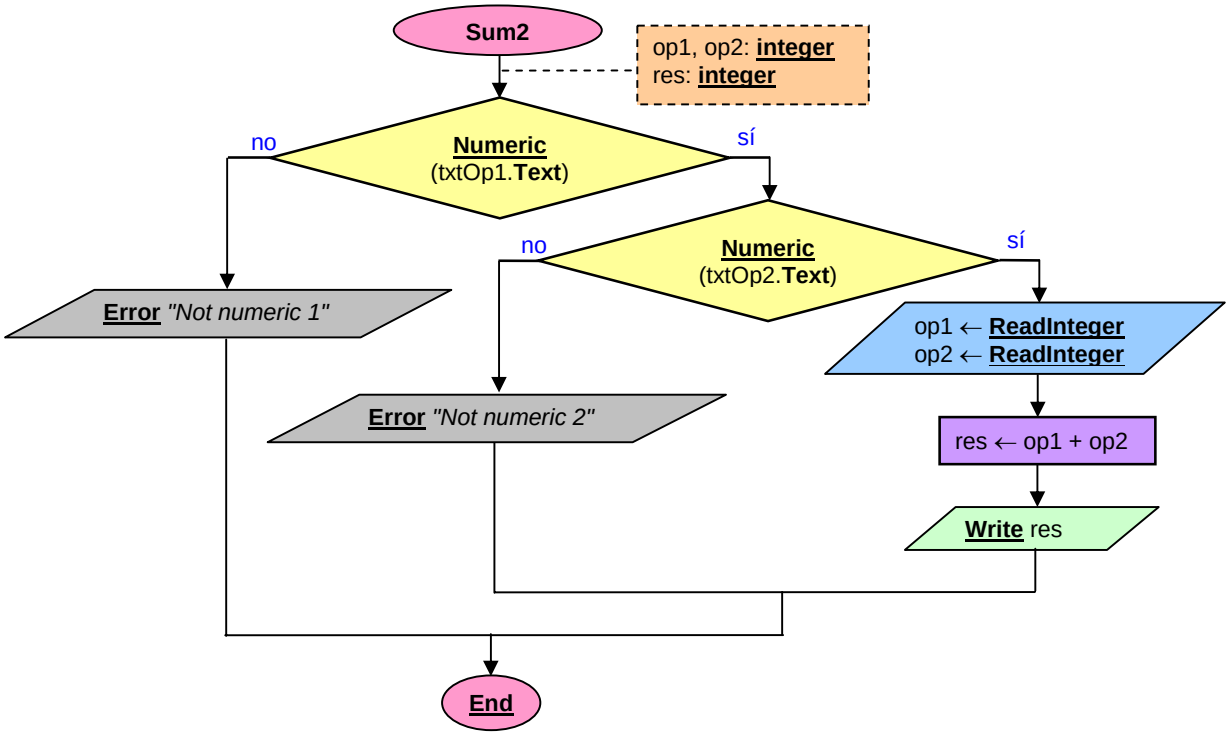

Figure 3.6 Flowchart of the sum with two checks.

```
Sub cmdSum2_Click()
  Dim op1 As Integer, op2 As Integer
  Dim res As Integer

  If IsNumeric(txtOp1.Text) Then
    If IsNumeric(txtOp2.Text) Then
      op1 = CInt (txtOp1.Text)
      op2 = CInt (txtOp2.Text)
      res = op1 + op2
      txtRes.Text = CStr (res)
    Else
      MsgBox ("Operand 2 is incorrect")
    End If
  Else
    MsgBox ("Operand 1 is incorrect")
  End If
End Sub
```

Figure 3.7 VB code for the sum with two checks.

Adder with check (3)

For our two previous sections we have utilized a reading model based on text boxes. To carry out a similar checking using an **InputBox** model we need to read the numbers into string variables, carry out the checking on these and if they have adequate numeric values obtain these on integer variables. The flowchart of this model can be seen in Figure 3.8.

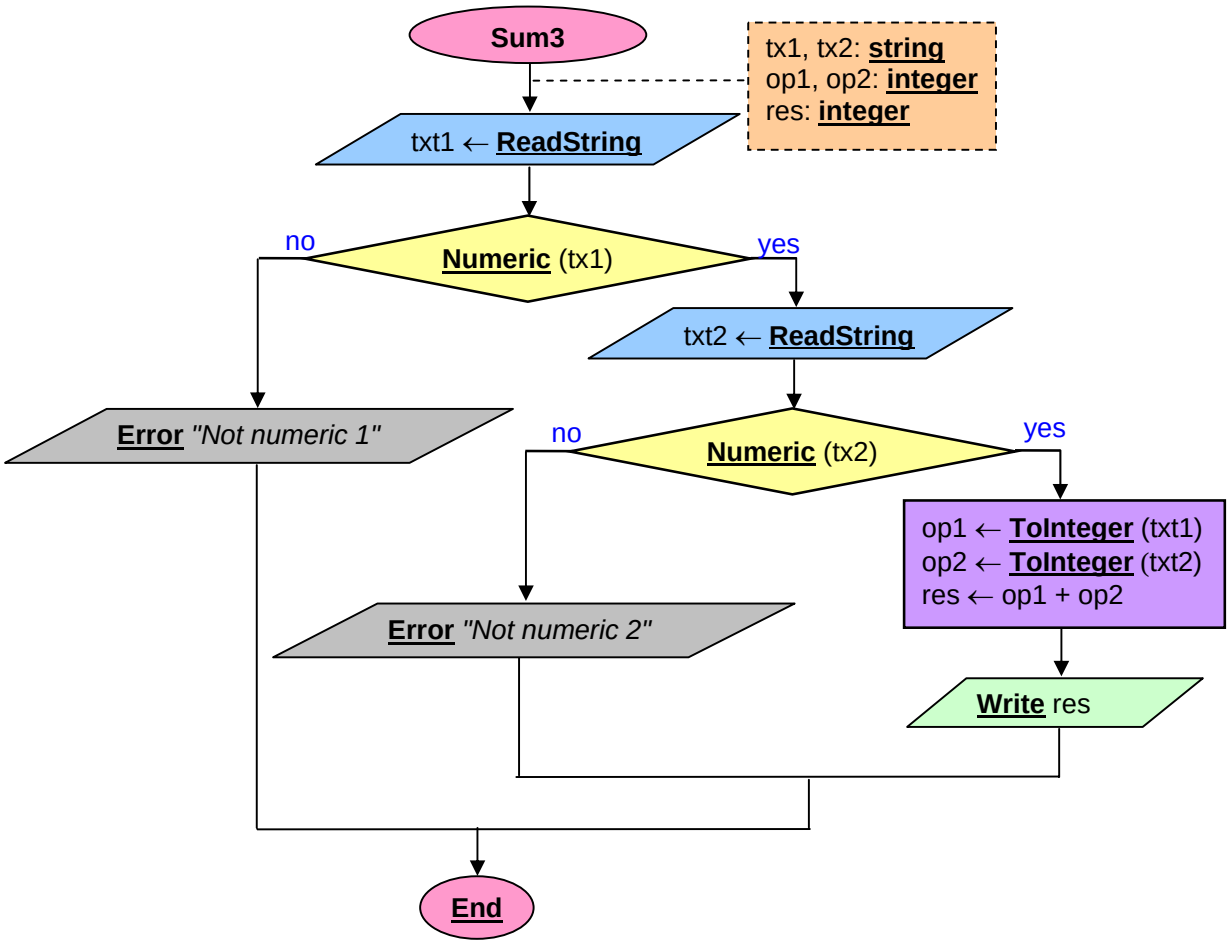

Figure 3.8 Flowchart of the sum with two checks using **InputBox**.

```
Sub cmdSum3_Click()
  Dim str1 As String, str2 As String
  Dim op1 As Integer, op2 As Integer
  Dim res As Integer
  str1 = InputBox ("Introduce first number")
  If IsNumeric(str1) Then
    str2 = InputBox ("Introduce second number")
    If IsNumeric(str2) Then
      op1 = CInt (str1)
      op2 = CInt (str2)
...
```

Figure 3.9 Portion of the adding subprogram using **InputBox**.

Exercise 3.1: Calculator 2

Design the flowchart, the **interface** and **implement** the VB calculator shown in Figure 3.10.

Interface

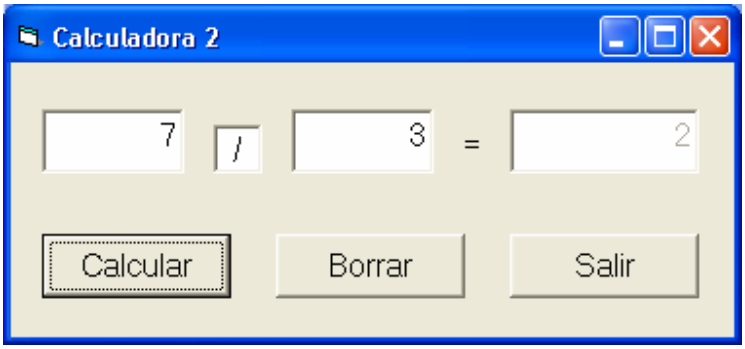

Figure 3.10 Interface of the second calculator model.

Operation

1. This application can be used to calculate the result of the operations **addition** (+), **subtraction** (–), **multiplication** (*), **quotient** of the integer division (/) and **remainder** (%). Note that in Visual Basic the quotient of the integer division is represented by a **backslash** (\) and the remainder by the **`Mod`** keyword.
2. The values introduced in the text boxes will be controlled to ensure that they are correct (numeric) and to prevent division by zero.
3. There will be an additional control to verify that the operator is one of the given five ones, namely: '+', '–', '*', '/' or '%'.
4. The text box for the results is not editable (**`Enabled`** property).
5. The command buttons will act as follows:
   - **Calculate button**: carry out the operation.
   - **Erase button**: remove the contents of the text boxes.
   - **Quit button**: finish the program.

Exercise 3.2: program to calculate areas

Design the **interface** and **implement** the VB area calculator shown in Figures 3.11, 3.12 and 3.13. There will be a single form and we will play making visible and invisible the different controls. Note that it is not necessary to draw the flowchart. All data are real.

Interfaces

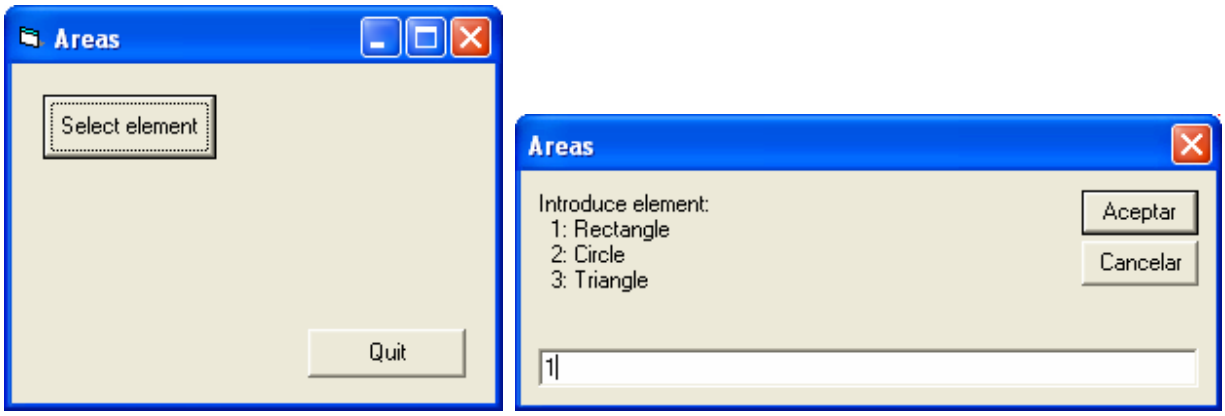

Figure 3.11 Initial window and **InputBox** reading an option

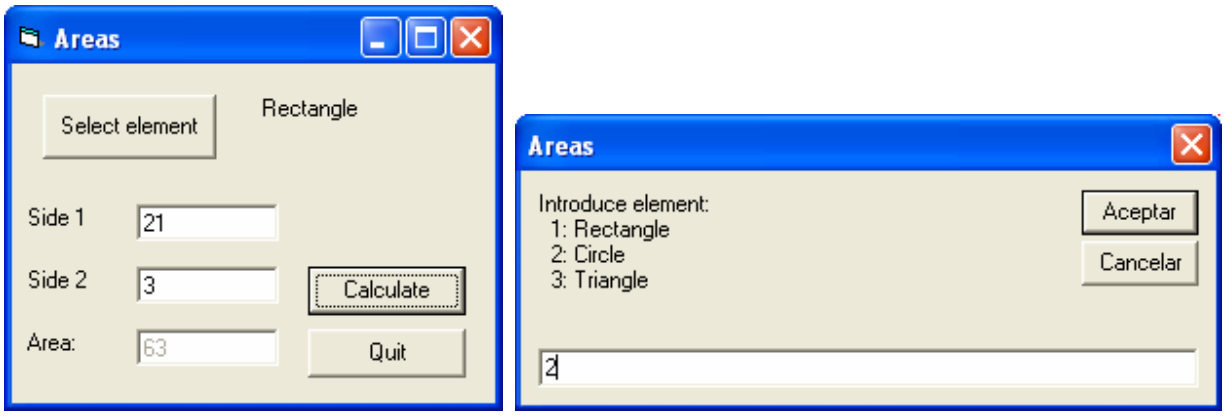

Figure 3.12 Visible controls for a rectangle and another **InputBox**.

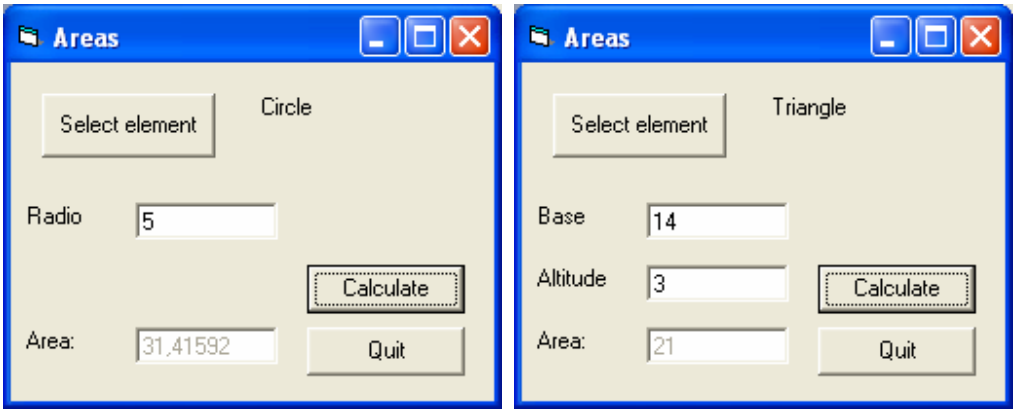

Figure 3.13 More interfaces of the area calculator.

Operation

1. This application can be used to calculate the area of a rectangle, a circle or a triangle. In each of the cases it will ask for the necessary data, hiding those objects that are not necessary for the given figure (setting to **`False`** the **`Visible`** property) and modifying the labels (**`Caption`** property) when necessary. The text box for the results is not editable (**`Enabled`** property set to **`False`**).

Note: as in previous exercise you must control that the values introduced in the text boxes are correct (i.e. numeric).

2. The buttons in the program behave as follows:

   - **"Select element" button:** launch a message box (**`MsgBox`**) with a list of figures to calculate the area. To cut lines you can use the constant **`vbCrLf`** as an end of line. Verify that the read option is valid (1, 2 or 3). Visualize the pertinent labels hiding those objects that are not adequate.
   - **"Calculate" button**: carry out the operation.
   - **"Quit" button:** Terminate the execution of the program.