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

Fundamentals of Computer Science
2010-2011
Laboratory 4
Conditional structures (2)

eman ta zabal zazu

Objectives:

- Design the flowchart of programs with **conditional sentences**
- **Implement** VB programs with **conditional sentences** (`If`, `If`-`Else`, `If`-`ElseIf`-`Else`).

Program to solve first and second degree equations

Interface

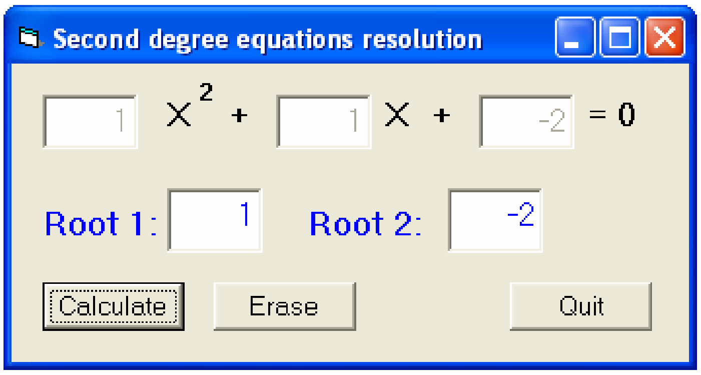


Figure 4.1 Equation calculator controls.

Operation

1. When launching the program the **result controls** (in blue) are **invisible**
2. When we click on the Calculate button the program must:
   - **Read** the value of the **a, b** and **c** coefficients of the equation
   - Calculate the **roots** of the equation
   - Block the fields to prevent from modifying them therefore losing coherence.
3. When we click on the Erase button we must remove the contents of the text boxes and get back to the original state, that is, with the results controls invisible
4. When we click on the Quit button the program will finish.

Cases study

For the resolution of a second degree (or quadratic) equation given their three coefficient we can distinguish four cases: an impossible or trivial solution (when $a=0$ and $b=0$), an equation of first degree (when $a=0$ and $b\neq0$), an equation of second degree (when $a\neq0$ and $b\neq0$) with real solutions (when the discriminant $d=b^2-4ac$ is positive or null) or with imaginary (complex) solutions (when the discriminant is negative). These cases are summarised in Table 4.1.

General formula	1st degree	2nd degree real	2nd degree imaginary
$r = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$	$r = \frac{-c}{b}$	$r_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a}$	$r_1 = \frac{-b}{2a} + \frac{\sqrt{4ac - b^2}}{2a} i$
		$r_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a}$	$r_2 = \frac{-b}{2a} - \frac{\sqrt{4ac - b^2}}{2a} i$
Particular case	$a = 0$	$d = b^2 - 4ac \geq 0$	$d = b^2 - 4ac < 0$

Table 4.1 Cases study.

Flowchart

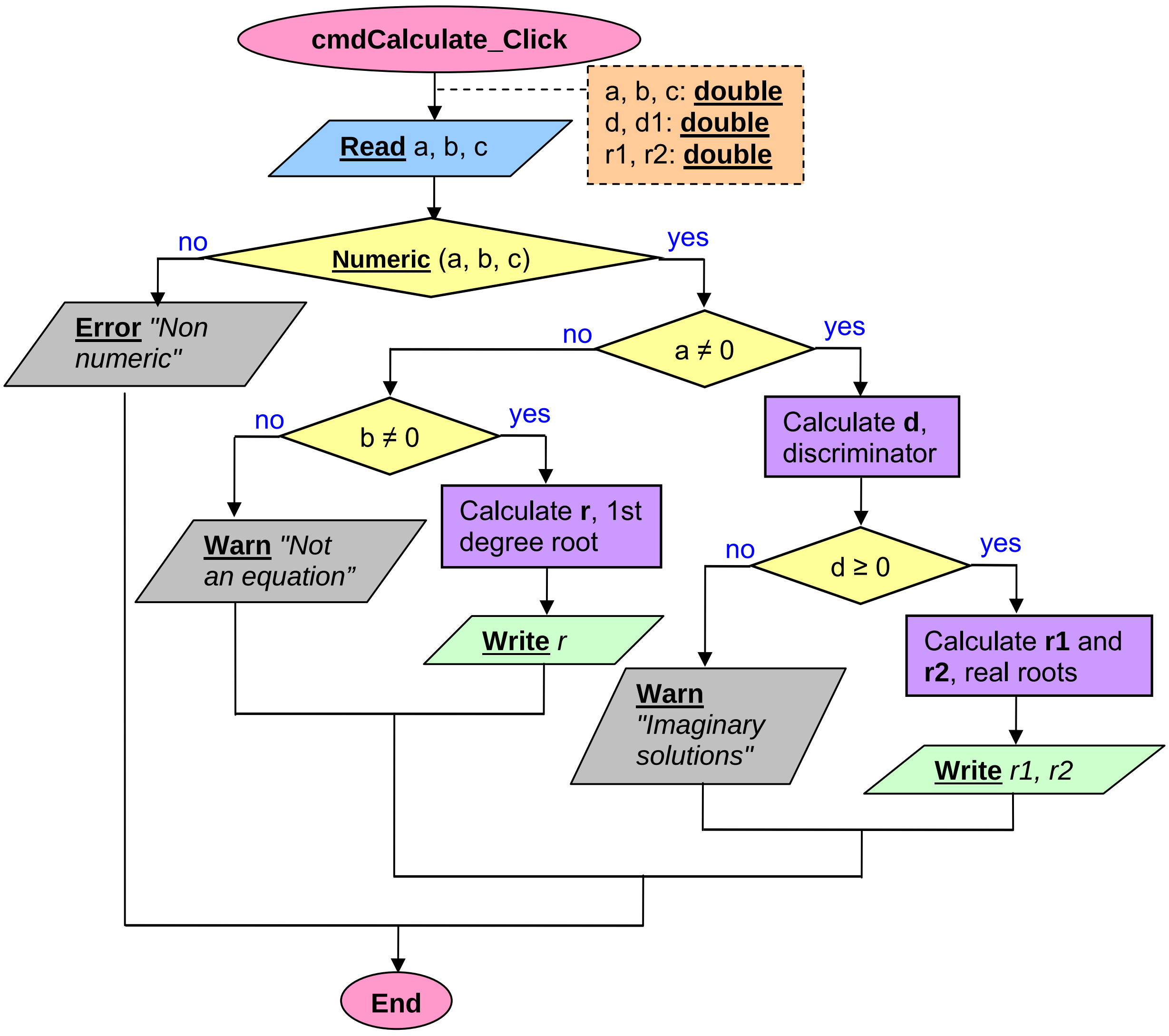


Figure 4.2 Equation resolution flowchart.

Steps

1) We create the controls of type and shape given in Figure 4.1. We only need to give a name to those controls that are to be recognised in the program to read or modify their properties. In Figure 4.3 we propose such names in red.

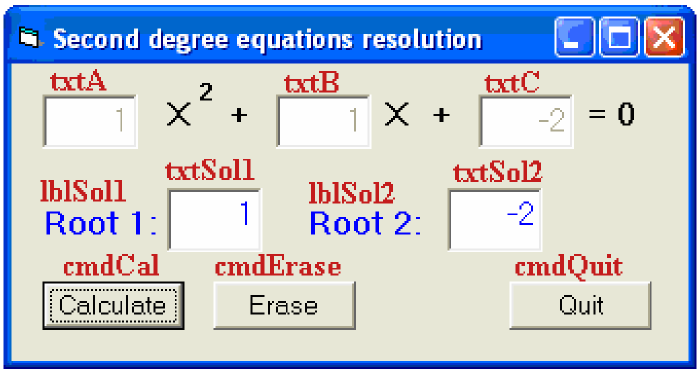


Figure 4.3 Names of the controls in the equations calculator.

2) We set to **False** the **`Visible`** property for those objects part of the solution, that is, `lblSol1`, `txtSol1`, `lblSol2` y `txtSol2`. This way, when we execute the program the first visualisation of the windows will be similar to that of Figure 4.4.

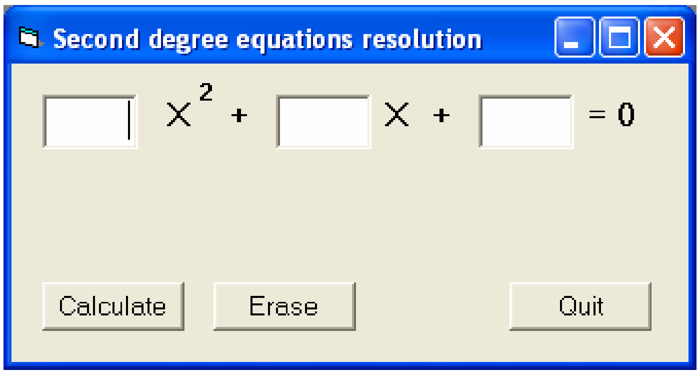


Figure 4.4 Program just after launch.

3) Code for the Calculate button: we must control the validity of the coefficients (they must be numeric) and depending on the values introduced by the user we determine the type of equation and calculate the results. Let's see the code step by step.

   i) We **declare** the variables for the coefficients **a**, **b** and **c** as **Double** (real), as well as the rest of the variables specified in the algorithm. We then control that these values are numeric by means of the **`IsNumeric`** function and if so **read** their values to the corresponding variables through the **CDbl** instruction. The corresponding code can be seen in Figure 4.5.

```
Sub cmdCalculate_Click()
  Dim a As Double, b As Double, c As Double
  Dim d As Double, d1 As Double ' Discriminant and its square root
  Dim r1 As Double, r2 As Double
    ' Obtain the values of the coefficients
  If IsNumeric(txtA.Text) And _
     IsNumeric(txtB.Text) And _
     IsNumeric(txtC.Text) Then
      ' The values must be numeric
    a = CDbl (txtA.Text)
    b = CDbl (txtB.Text)
    c = CDbl (txtC.Text)
      ' Block coefficient text boxes
    txtA.Enabled = False
    txtB.Enabled = False
    txtC.Enabled = False
. . . (1)
  Else
    MsgBox "Error: non-numeric coefficients"
  End If

End Sub ' End of cmdCalculate_Click
```

Figure 4.5 Simplified code for Calculate button.

ii) Having read the value of the variables, following the algorithm, we determine if it is a **second degree** equation, **first degree** or if **it is not** an equation. The code for the corresponding conditional instructions is shown in Figure 4.6. Pay special attention to the indentation because it especially affects the readability of the program.

```
    txtC.Enabled = False (1)
    If a <> 0 Then ' Second degree equation
. . . (2)
    Else
      If b <> 0 Then ' First degree equation
. . . (3)
      Else ' Not an equation
        MsgBox "Error: it is not an equation"
      End If
    End If
Else
```

Figure 4.6 Code with the conditions for the different cases.

iii) For the second degree equations (coefficient **a** ≠ 0) we calculate the **discriminant** (**d** variable). If this is **null** we will have **two identical real** roots; if it is **positive** we will have **two different real** roots and if it is **negative** we will have **two different imaginary** roots. In this latter case we will just indicate this condition with a message. See code in Figure 4.7.

```
    If a <> 0 Then ' Second degree equation (2)
      d = b ^ 2 - 4 * a * c ' Discriminant calculation
      If d >= 0 Then ' Real solutions
. . . (4)
      Else ' Imaginary solutions
        MsgBox "Only imaginary solutions"
      End If
    Else
```

Figure 4.7 Code for the different cases depending on the discriminant.

iv) To calculate the real roots we need to obtain the square root through the Visual Basic **`Sqr`** function. This way, we obtain the roots and we display them, setting up texts and making visible controls as appropriate. This code is shown in Figure 4.8.

```
    If d >= 0 Then ' Real solutions (4)
      d1 = Sqr(d) ' Square root
      r1 = (-b + d1) / (2 * a)
      r2 = (-b - d1) / (2 * a)
      txtSol1.Text = CStr (r1)
      txtSol2.Text = CStr (r2)
      lblSol1.Caption = "Root 1:"
      lblSol1.Visible = True
      txtSol1.Visible = True
      lblSol2.Caption = " Root 2:"
      lblSol2.Visible = True
      txtSol2.Visible = True
    Else ' Imaginary solutions
```

Figure 4.8 Code to set up controls for real solutions.

v) Finally, when the **a** coefficient is null we face a linear equation, with only one root. With this we finish with the code associated with the Calculate button. The code is shown in Figure 4.9.

```
    If b <> 0 Then ' First degree equation (linear) (3)
      r1 = -c / b
      txtSol1.Text = CStr (r1)
      lblSol1.Caption = "Root:"
      lblSol1.Visible = True
      txtSol1.Visible = True
    Else
```

Figure 4.9 Code to set up controls for linear equations.

4) Code for the Erase button: we need to make the solution controls invisible: `lblSol1`, `txtSol1`, `lblSol2` y `txtSol2`, as we have done in the development environment, setting the **`Visible`** property **`False`**.
Additionally we unblock the text boxes for the coefficients **a**, **b** and **c**, emptying their contents. The code for these actions is provided in Figure 4.10.

```
Sub cmdErase_Click()
    ' Make invisible objects for the solution
  lblSol1.Visible = False
  lblSol2.Visible = False
  txtSol1.Visible = False
  txtSol2.Visible = False
    ' Unblock A-B-C
  txtA.Enabled = True
  txtB.Enabled = True
  txtC.Enabled = True
  txtA.Text = ""
  txtB.Text = ""
  txtC.Text = ""
End Sub
```

Figure 4.10 Code for the Erase button.

5) Code for the Quit button: As in previous examples.

Exercise 4.1: Roller program

Design the **flowchart**, the **interface** and the **Visual Basic** program of the **roller** in Figure 4.11 without referring to the text in the program. It we change the text in the interface the roller must work with the new text.

Interface

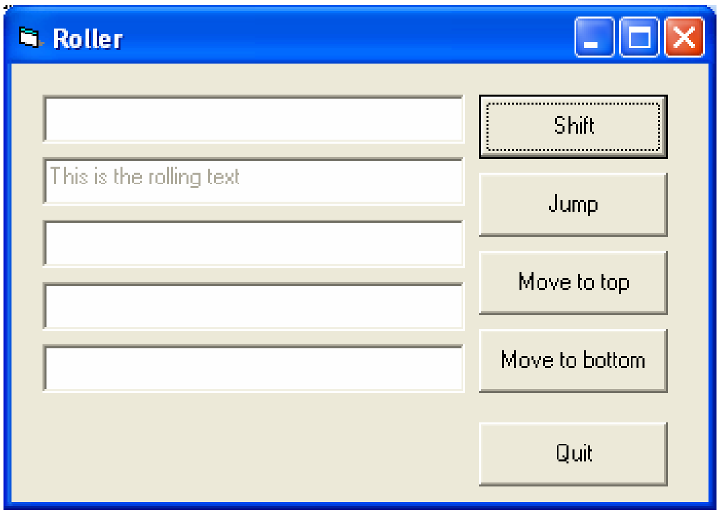


Figure 4.11 Objects present in the roller interface.

Operation

1. This program contains 5 **non-editable** text boxes (**`Enabled`** property set to **`False`**). One of them has a text, for example, “*This is the rolling text*”. The different buttons make “roll” the text.
2. Shift button: The text shifts to the following position. If the text is in the last text box it will go to the first text box.
3. Jump button: The text goes in a circular manner always leaving a gap or empty text box in between the previous text box and the new position. When it reaches the last position it returns to the beginning
4. Move to top button: moves the text to the first position (top).
5. Move to bottom button: moves the text to the last position (bottom).

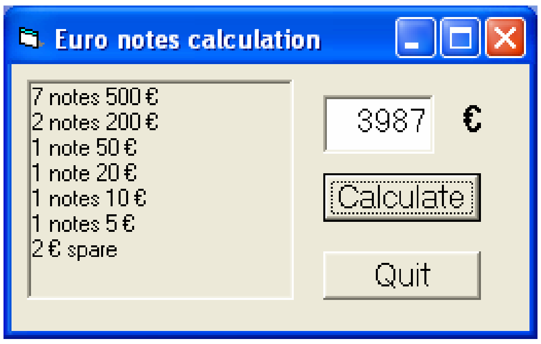
Hint

Use conditional sentences to determine where the text can be found and then decide the new location, depending on the semantics of the button. Copy the text from the origin to the destination and empty the text box of origin.

Exercise 4.2: Calculation of Euro notes (2)

Design the **interface** and implement the **Visual Basic program** of the **Euro note calculation** program of Figure 4.12. The fundamentals are the same as in exercise 2.4 but we only show relevant information (non-null values) and we distinguish the case of one note and several. The notes considered are 500, 200, 100, 50, 20, 10, 5 and spare euros.

Interface



```
7 notes 500 €
2 notes 200 €
1 note 50 €
1 note 20 €
1 notes 10 €
1 notes 5 €
2 € spare
```

3987 €

Figure 4.12 Interface for the Euro notes calculation program.

Exercise 4.3: Calculation of the following day

Design the **interface** and **implement** the Visual Basic program for the date calculator in Figure 4.13.

Interface

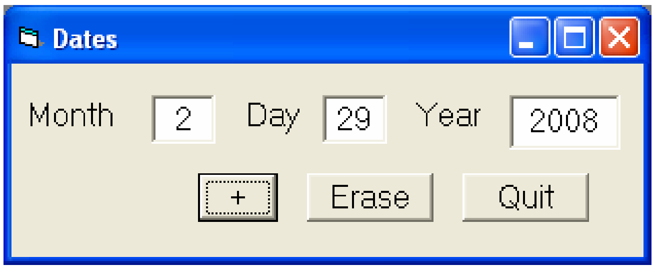


Figure 4.13 Interface for the date calculator.

Operation

The program works with dates (try executable demo).

1. + button: Controls that the introduce date is correct (four digits for the year) and calculates the following date, displaying it on the corresponding text boxes.
2. Erase button: Cleans the contents of the text boxes.
3. Quit button: Finish the program.

Checks

1	Contents of **Month**, **Day** and **Year** fields is **numeric**
2	**Day correct**: greater than or equal to 1 and less than or equal to 31
3	**Month correct**: greater than or equal to 1 and less than or equal to 12
4	**Year correct**: greater than or equal to 1000 and less than or equal to 9999
5	**Date correct**: there is no **31st** of **February**, **April**, **June**, **September** or **November**
6	**Date correct**: there is no **30th of February**
7	**Date correct**: there is no **29th of February** when not a leap year[1]

Recommendation

Note that having incremented the day on a correct date if we get to an incorrect date, the day will be 1 and the month will need to be incremented. Similarly if the month becomes 13 it will be 1 and the year incremented by one. This checking is similar to the previous date checking.

[1] **Leap years** are those **divisible by 4 but not by 100 unless they are divisible by 400**. For example, 1996, 2000 and 2004 were leap years but 2100, 2200 and 2300 won't and again 2400 will be a leap year. We'll celebrate this year!

Quick reference tables

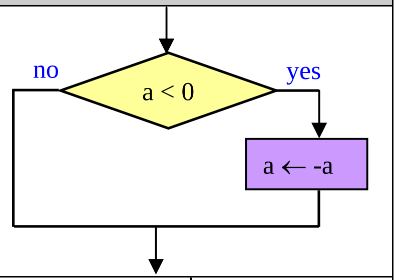
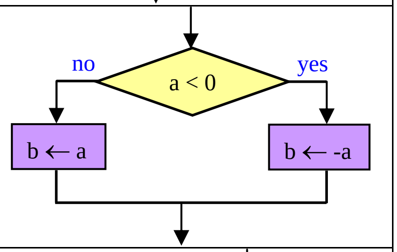
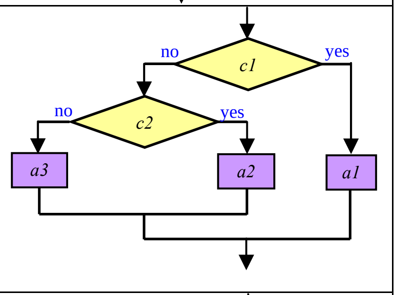
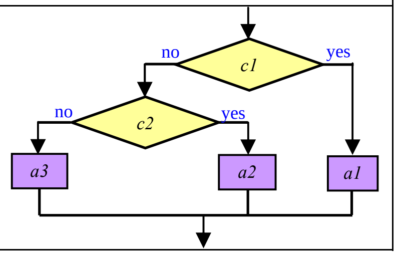
Conditional structures table		
Syntax	**Example**	**Flowchart**
`If` *c* `Then` ... `End If`	`If a < 0 Then` `a = -a` `End If`	
`If` *c* `Then` ... `Else` ... `End If`	`If a < 0 Then` `b = -a` `Else` `b = a` `End If`	
`If` *c1* `Then` *a1* `Else` `If` *c2* `Then` *a2* `Else` *a3* `End If` `End If`	`If a > b Then` `m = "a"` `Else` `If b > a Then` `m = "b"` `Else` `m = "="` `End If` `End If`	
`If` *c1* `Then` *a1* `ElseIf` *c2* `Then` *a2* `Else` *a3* `End`	`If a > b Then` `m = "a"` `ElseIf b > a Then` `m = "b"` `Else` `m = "="` `End If`	

Table 4.2 Syntax for conditional structures.

Visual Basic functions	
`IsNumeric`	Indicates if a string represents a numeric value
`Sqr`	Calculates the square root of a real number

Table 4.3 Visual Basic functions.