Subprograms

Fundamentals of Computer Science

2010-2011

Ismael Etxeberria Agiriano

08/11/2010

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

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea

Index

Subprograms

1. Calculation of the Cosine function

- ***Title***
  - Cosine
- ***Name***
  - PrgCosine
- ***Description***
  - VB program to read an angle in radians and calculate its cosine, utilizing the Taylor series with an error less than 0.000001.
- ***Observations***
  - Decomposition in functions
  - Design with and without functions
  - Top-down design, bottom-up implementation

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

1.1 Analysis Cosine

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

$$y = t_0 + t_1 + t_2 + t_3 + \ldots + t_\infty$$

$$y = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \ldots$$

Each t_i reduces the error in $|t_i|$

$$y = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

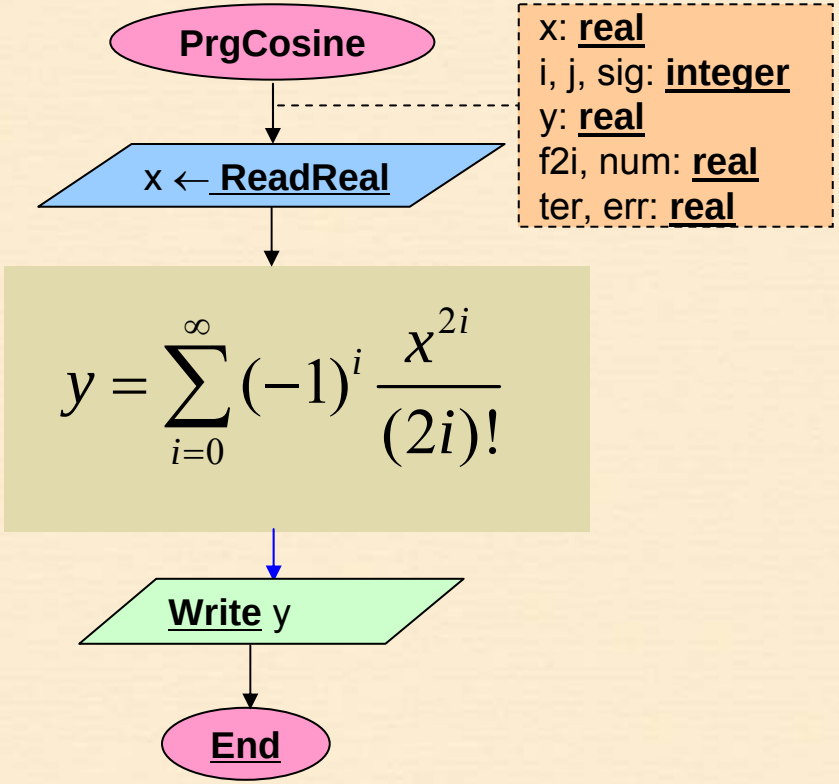
$$t_0 = -1^0 \cdot \frac{x^{2\cdot 0}}{(2\cdot 0)!} = 1$$

$$t_1 = -1^1 \cdot \frac{x^{2\cdot 1}}{(2\cdot 1)!} = -\frac{x^2}{2}$$

$$t_2 = -1^2 \cdot \frac{x^{2\cdot 2}}{(2\cdot 2)!} = \frac{x^4}{24}$$

$$t_3 = -1^3 \cdot \frac{x^{2\cdot 3}}{(2\cdot 3)!} = -\frac{x^6}{720}$$

1.2 Cosine program without subprograms

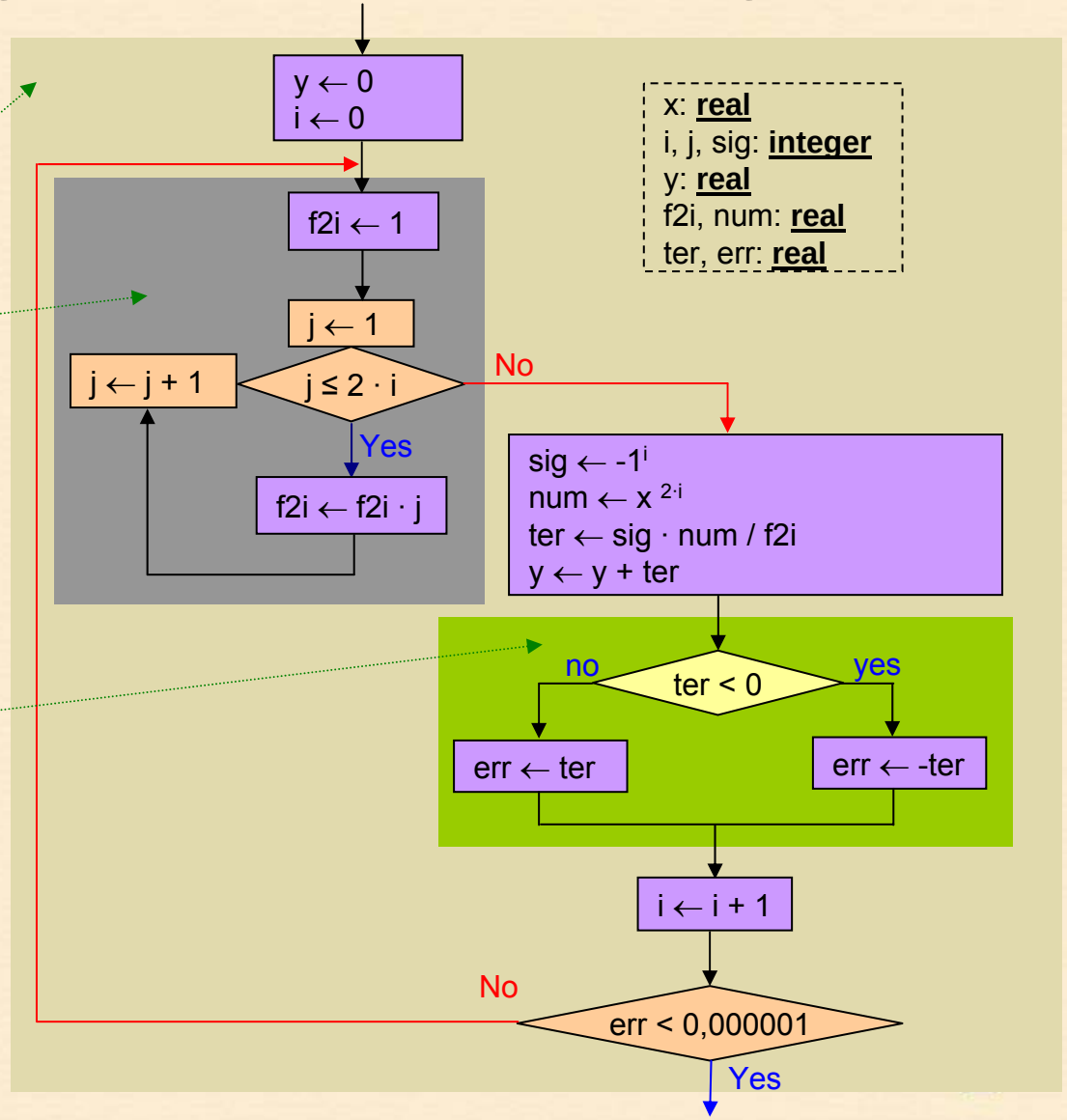


1.2 Cosine program without subprograms

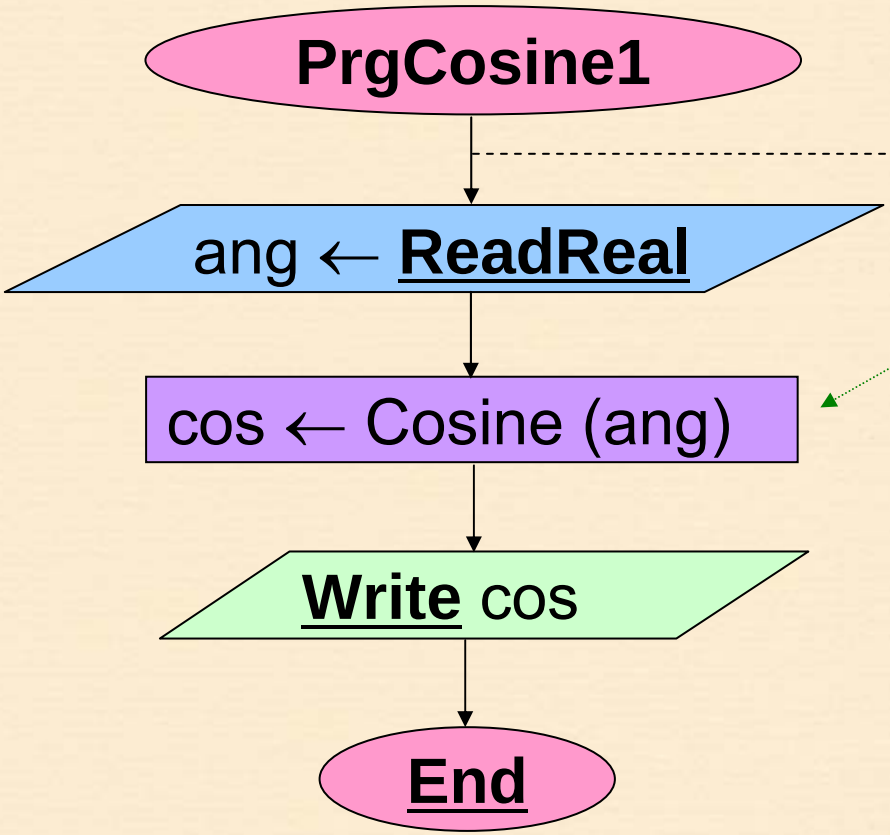
$$y = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

fact2i ← (2·i)!

err ← |ter|



1.3 Cosine program with subprograms



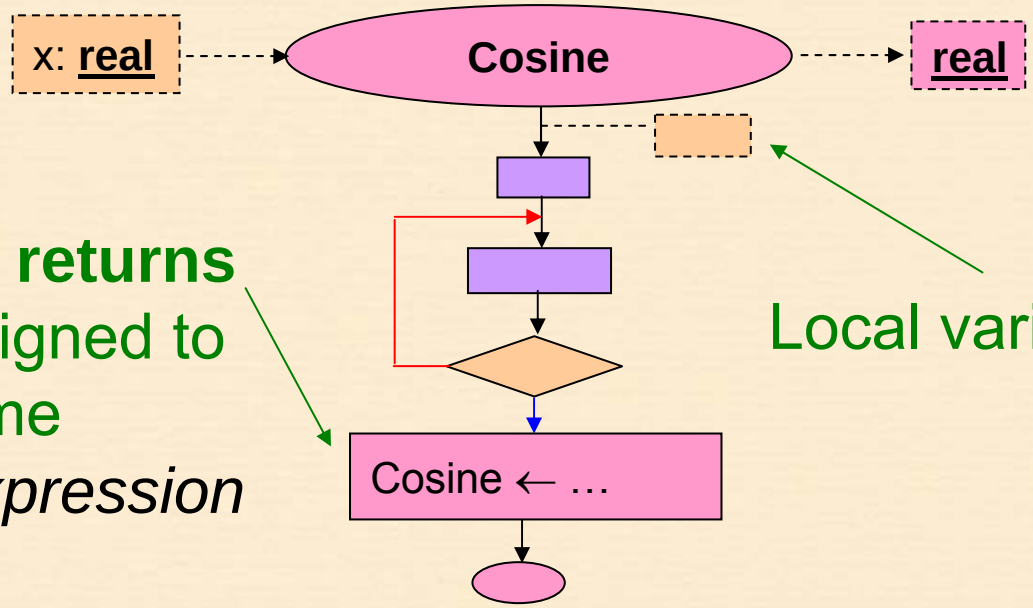
Call to the Cosine function with the **ang** value

ang is evaluated and this value is sent to the function

1.4 Cosine function interface

Input parameter: x
It is like a local variable initialized to the value given while calling

Type of the value returned by the function



The function **returns** a value, assigned to its name

Cosine ← *expression*

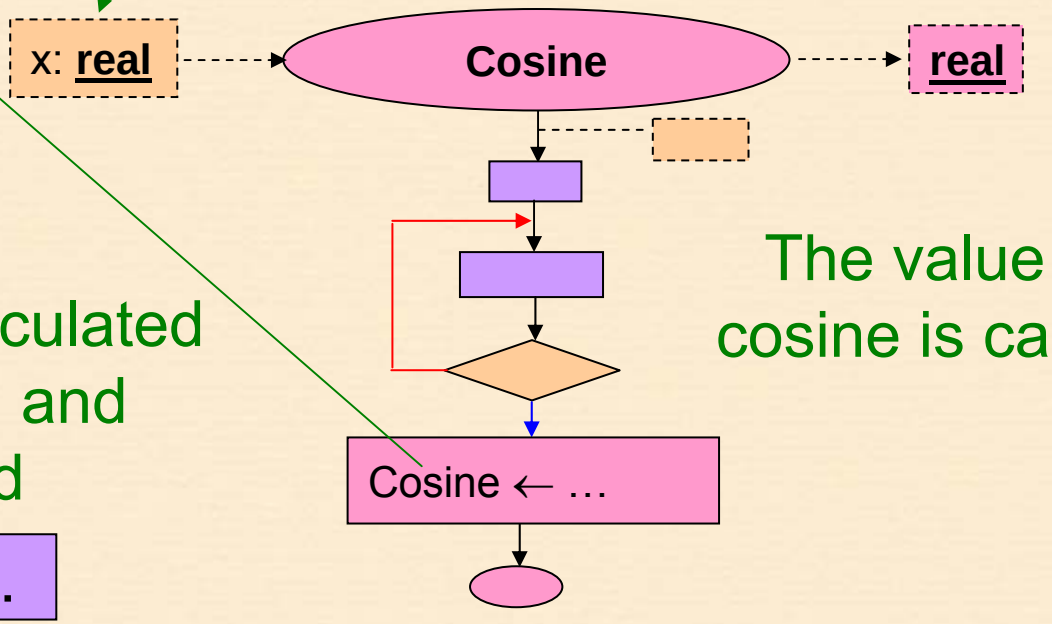
Local variables

1.5 Call to the Cosine function

cos ← Cosine (ang)

In the function call there is an implicit assignment

x ← ang



This value calculated is assigned and returned

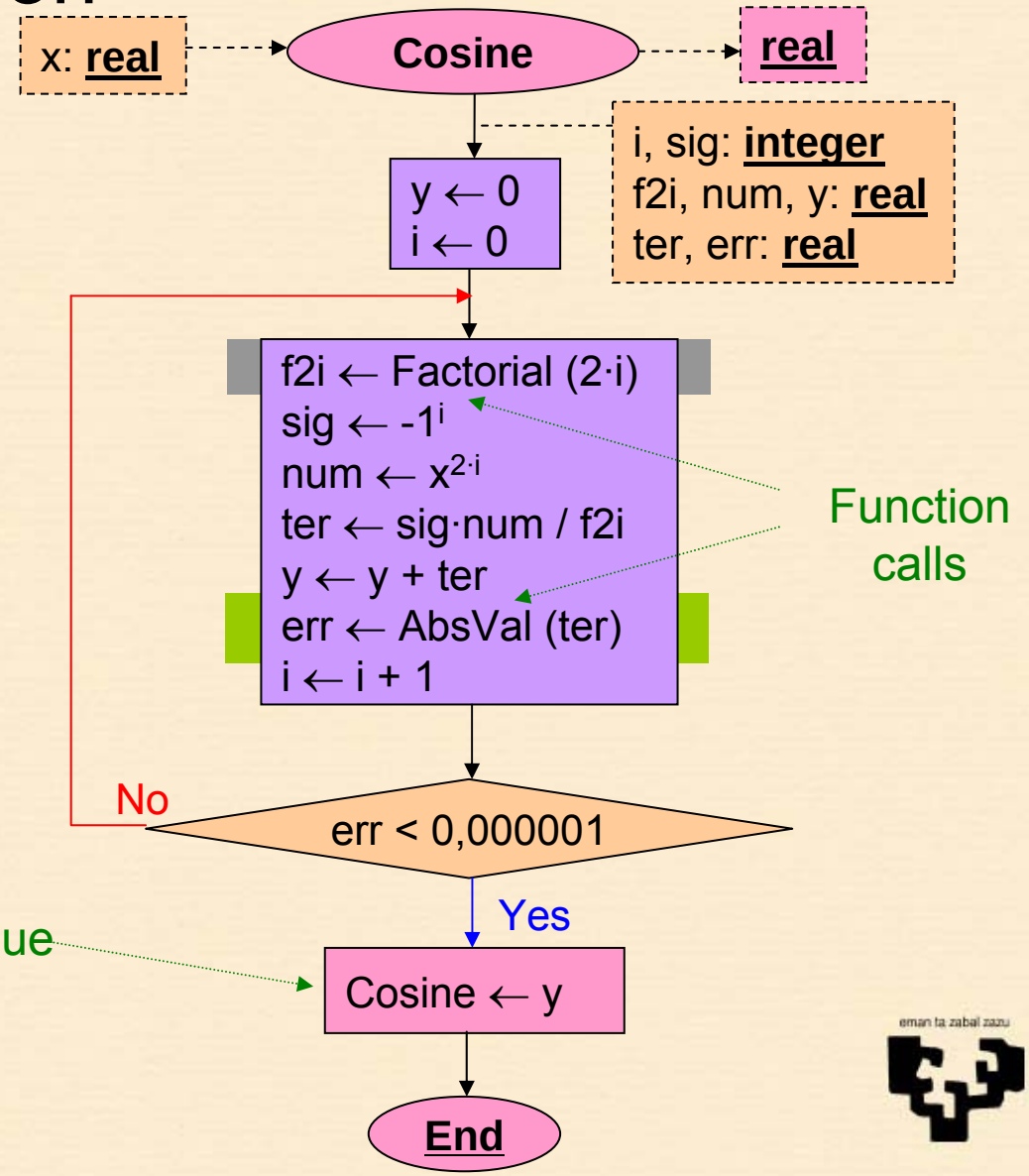
cos ← …

The value of the cosine is calculated

1.6 Cosine function

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

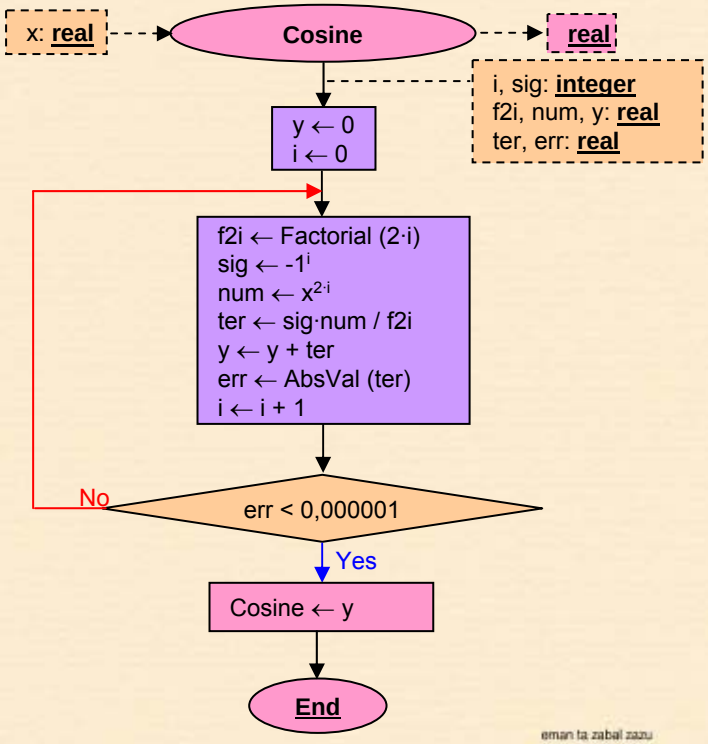
$$y = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$



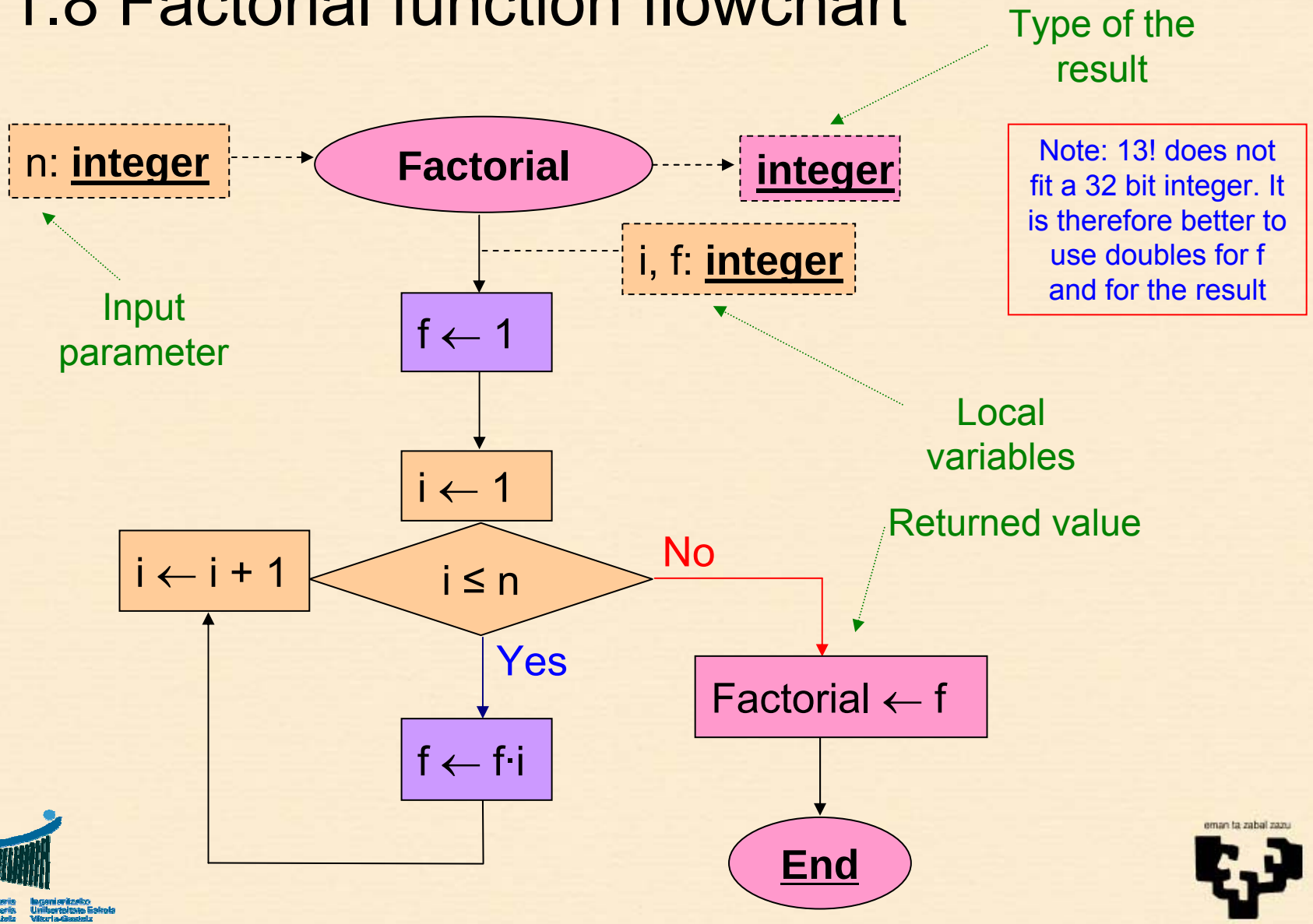
1.7 VB Cosine function

```
Function Cosine (ByVal x As Double)_
         As Double
  Dim i As Integer, sig As Integer
  Dim f2i As Integer, num ...
  y = 0
  i = 0
  Do
    f2i = Factorial (2*i)
    sig = (-1) ^ i
    ...
  Loop Until err < 0.000001
  Cosine = y
End Function
```

$$\cos(x) = \sum_{i=0}^{\infty} (-1)^i \frac{x^{2i}}{(2i)!}$$

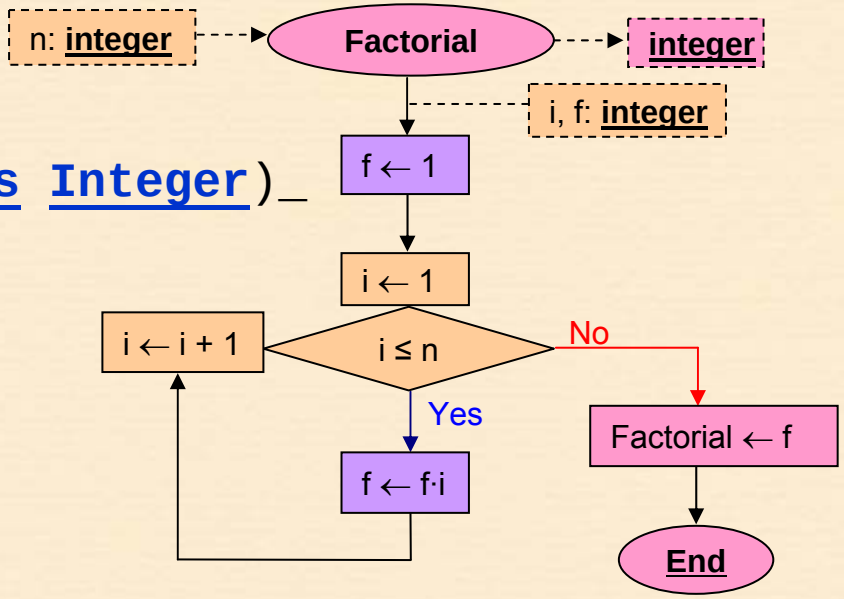


1.8 Factorial function flowchart



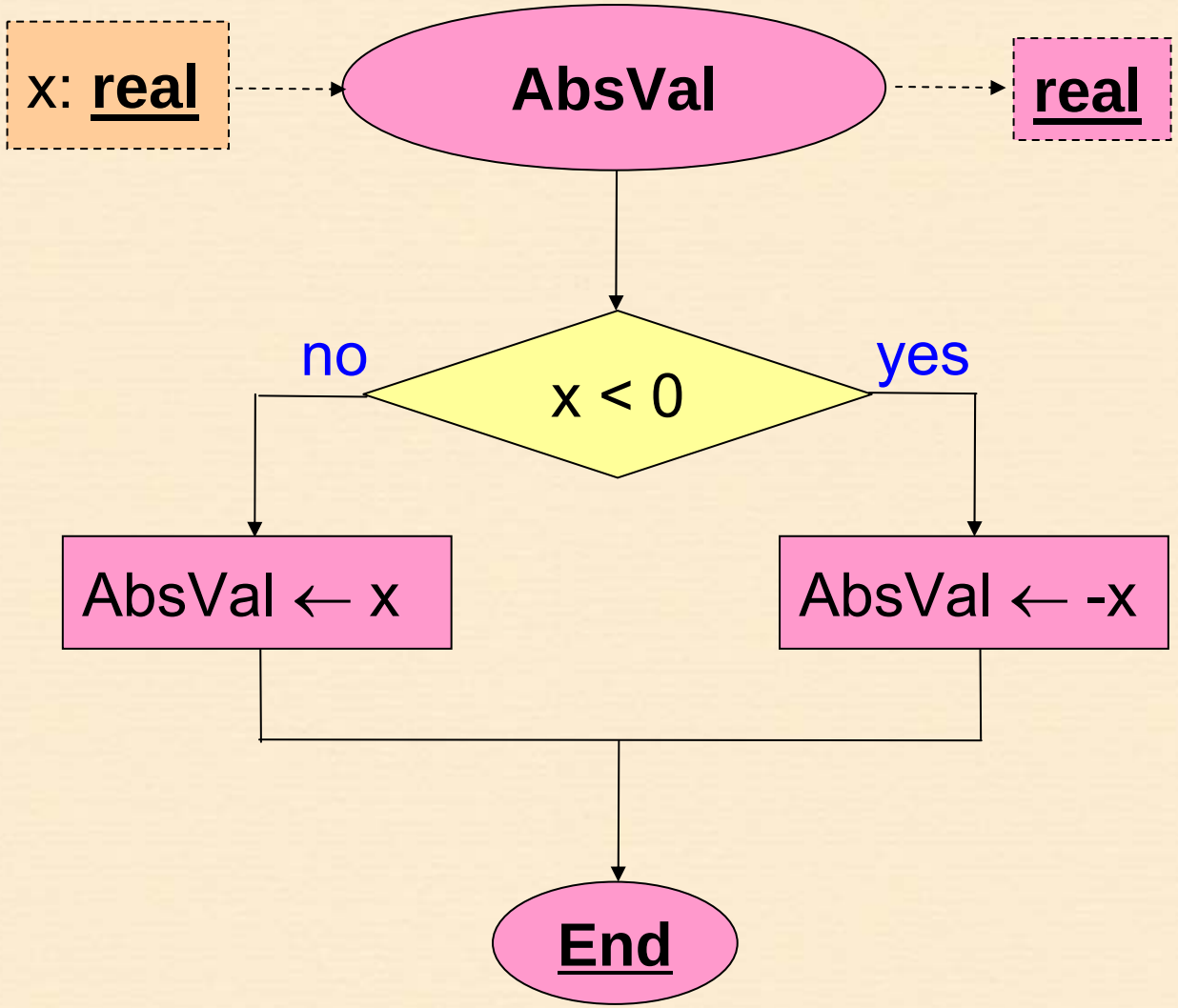
1.8 Factorial VB function

```
Function Factorial (ByVal n As Integer)_
                          As Double
  Dim i As Integer
  Dim f As Double
  f = 0
  For i = 1 To n Step 1
    f = f * i
  Next i
  Factorial = i
End Function
```

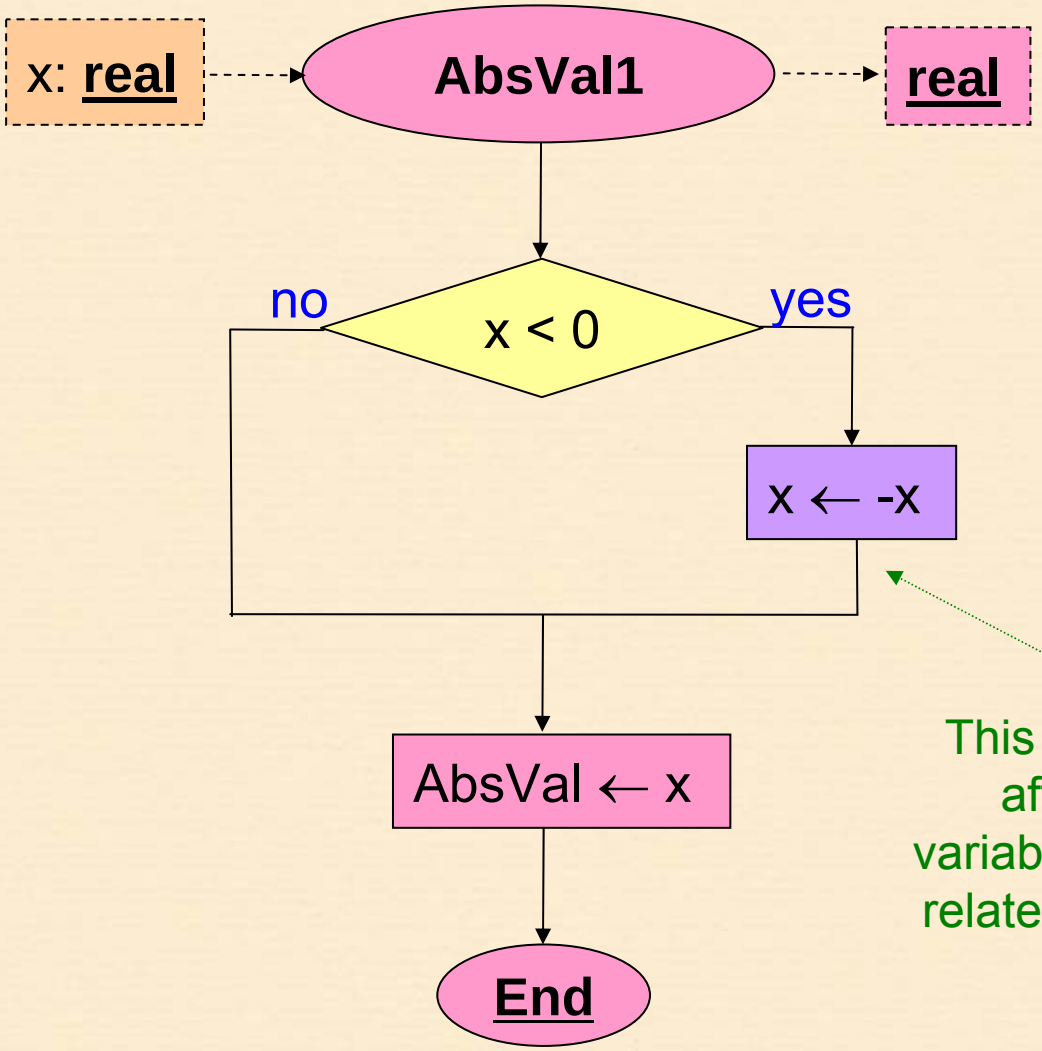


Note: we use doubles instead of integer

1.9 AbsVal function

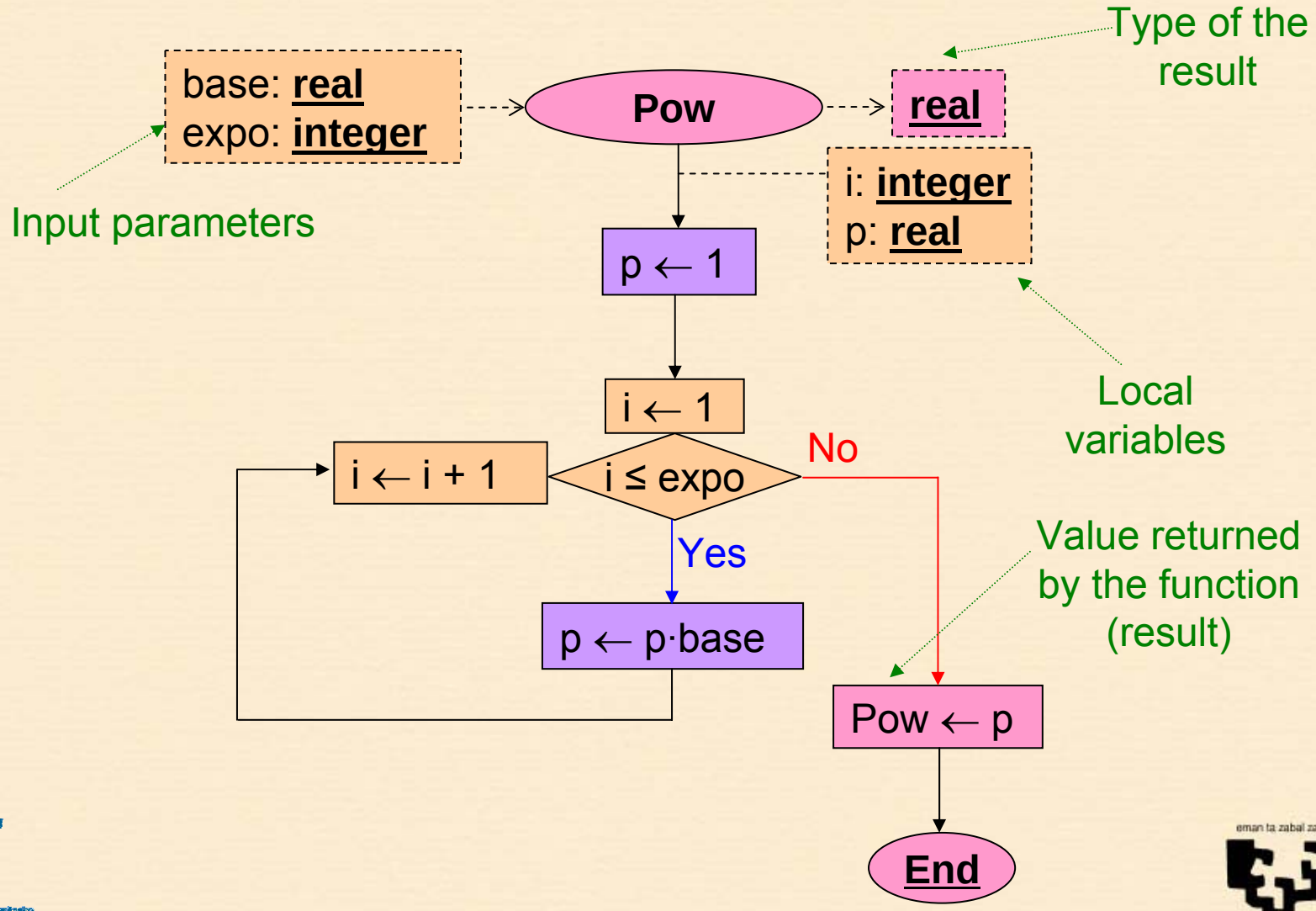


1.10 AbsVal1 function



This change does not affect the calling variable as it gets a non-related copy of its value

1.11 Pow function (not necessary in VB)



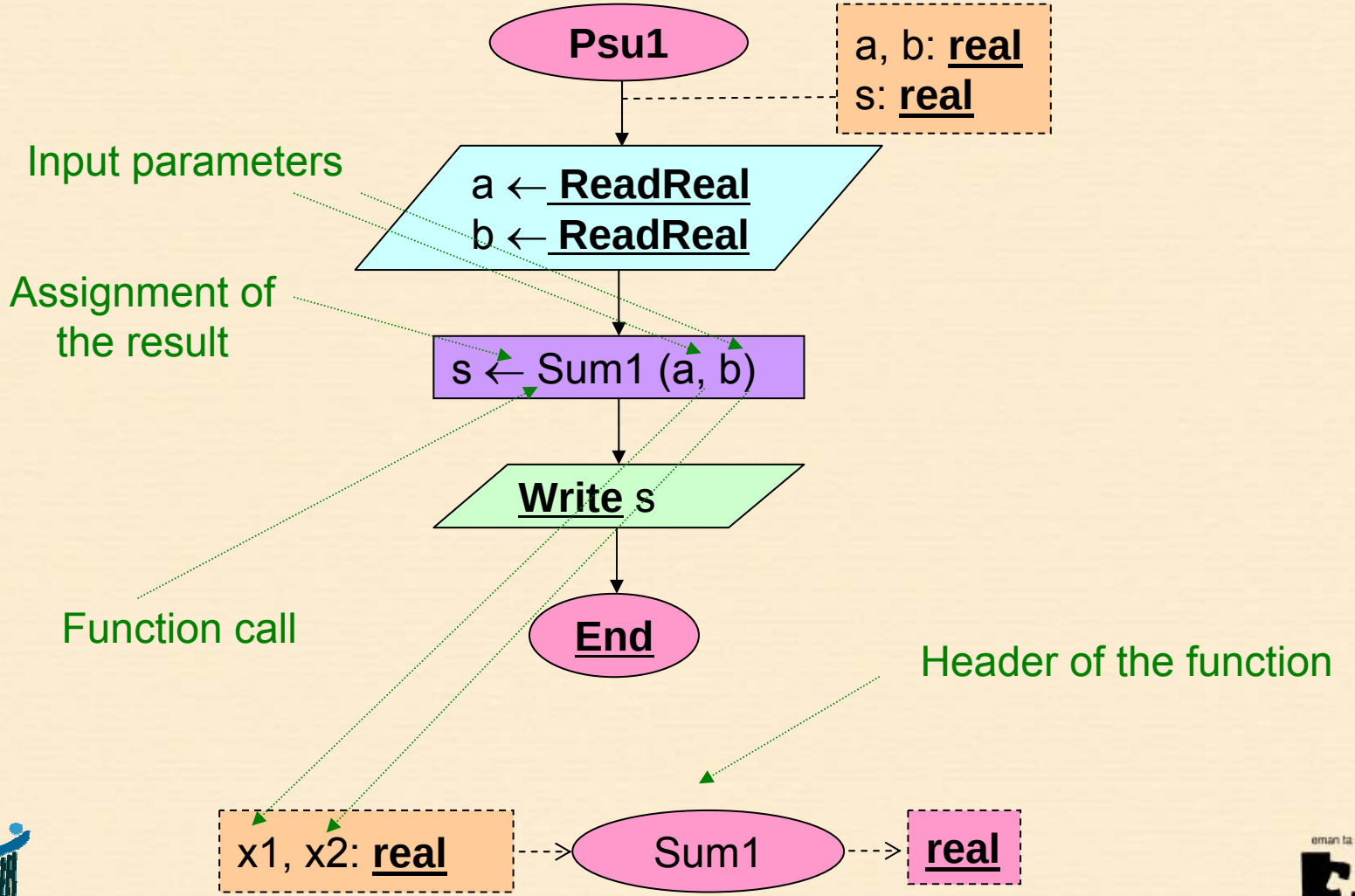
2. Sum – Parameter passing models

Description

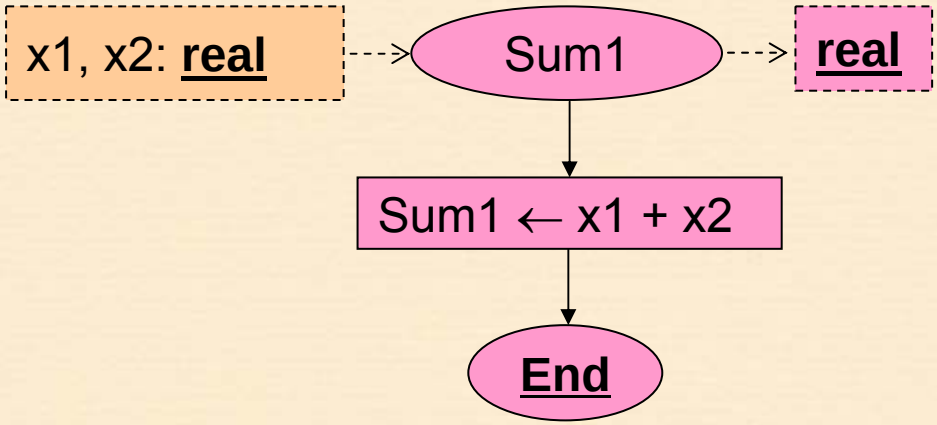
Design and implement a subprogram to sum two numbers

- ***Versions***
  1. **Function** with **two input** parameters and **returns** the result
  2. **Procedure** with **two input** parameters and **one output** parameter
  3. **Procedure** with **one input** parameters and **another input/output** parameter
- ***Observations***
  - Parameter passing **By Value** and **By Reference**
  - Procedures don't "return" values (but they may use output parameters)

2.1 Sum with a function: program and call



2.1 Sum1 function: Flowchart and VB



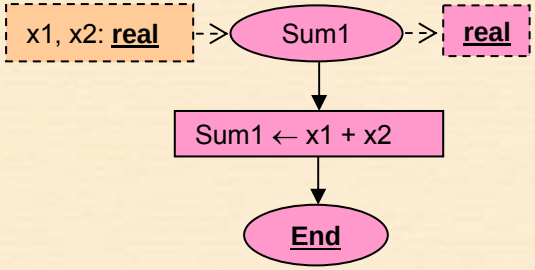

```
Function Sum1 (ByVal x1 As Double, ByVal x2 As Double)_
         As Double
  Sum1 = x1 + x2
End Function
```

2.1 Calls to the Sum1 function

Call with variables

s ← Sum1 (a, b)

```
...
s = Sum1 (a, b)
...
```

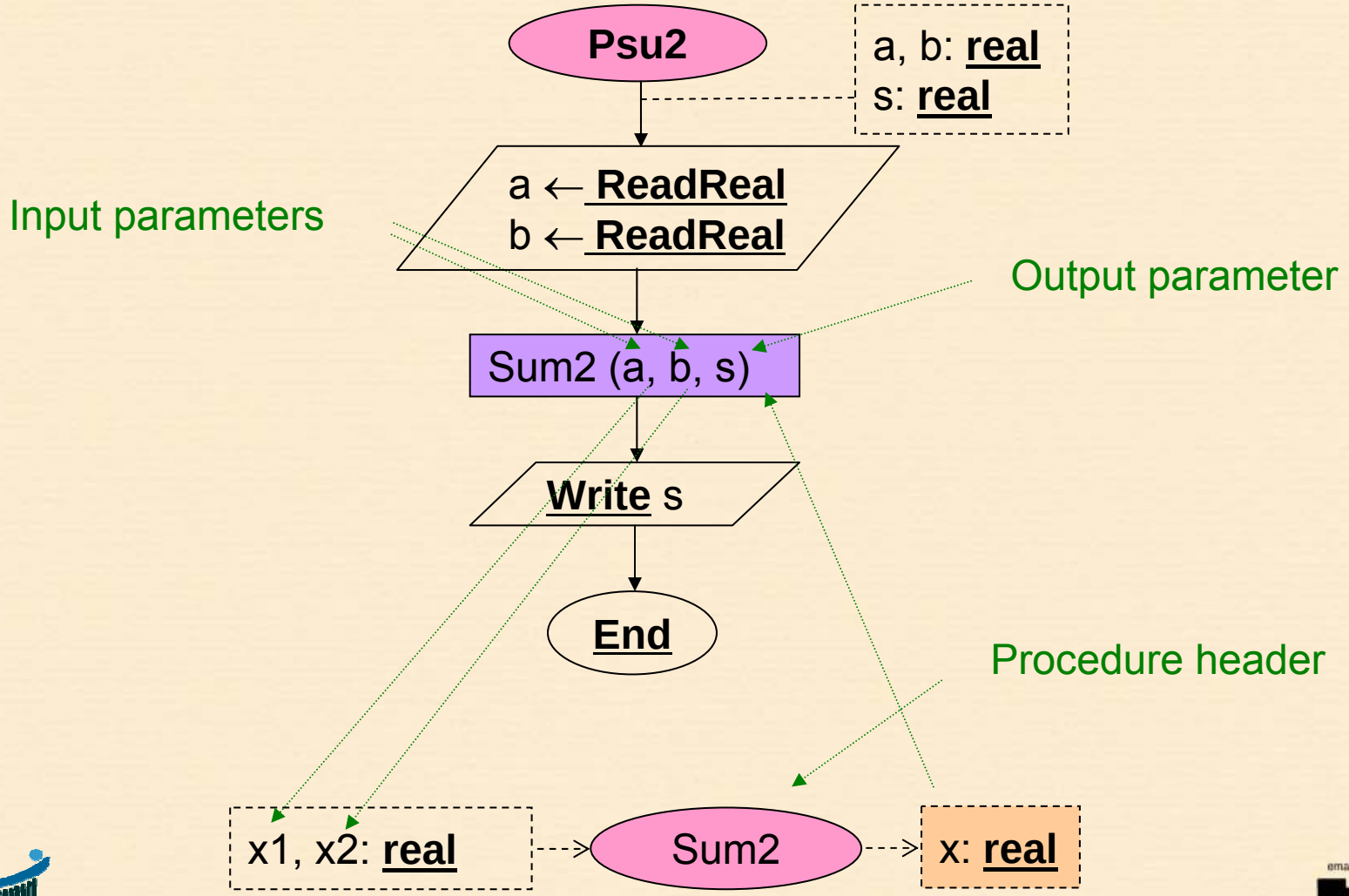


Call with literal constants

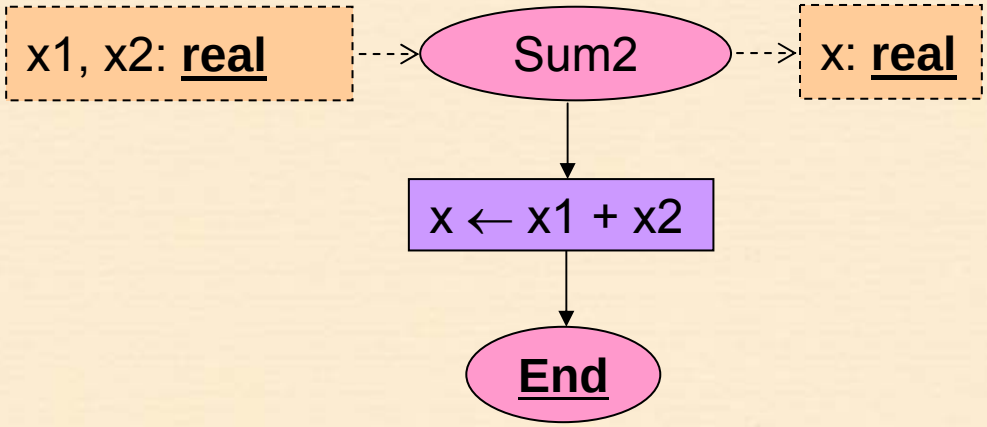
s ← Sum1 (5, 7)

```
...
s = Sum1 (5, 7)
...
```

2.2 Procedure 1: program and call



2.2 Sum2 procedure: flowchart and VB




```
Sub Sum2 (ByVal x1 As Double, ByVal x2 As Double, _
          ByRef x As Double)
  x = x1 + x2
End Sub
```

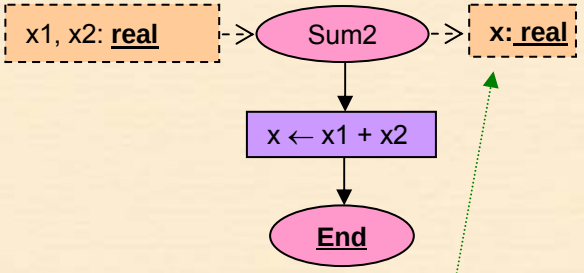
It may be removed

All modifications carried out on the variables passed by reference affect the calling variables, which may have the same name or different name

2.2 Calls to the Sum2 procedure

Call with variables

```
Sum2 (a, b, s)

...
Call Sum2 (a, b, s)
...
```

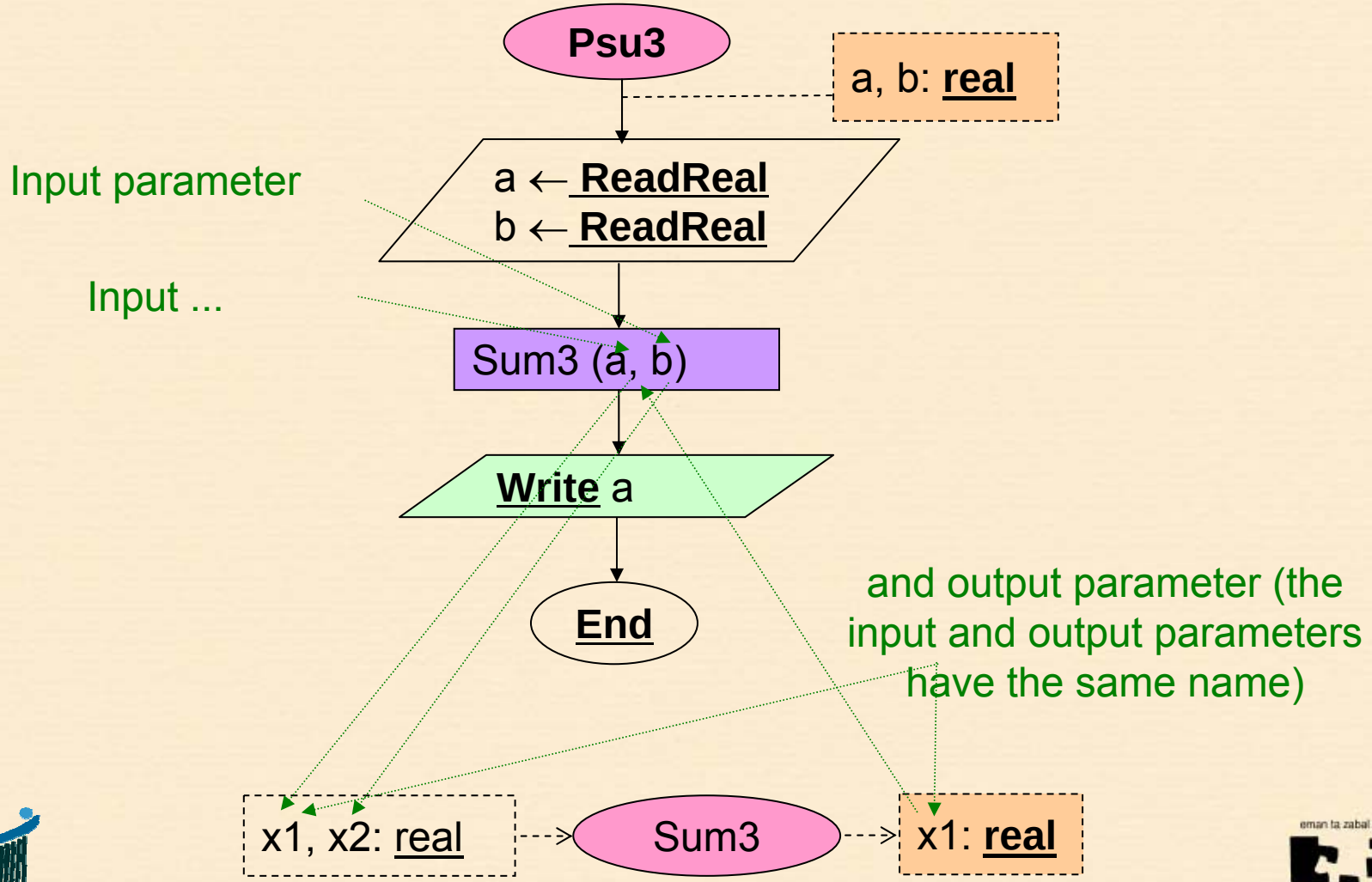


Call with literal constants (by value)

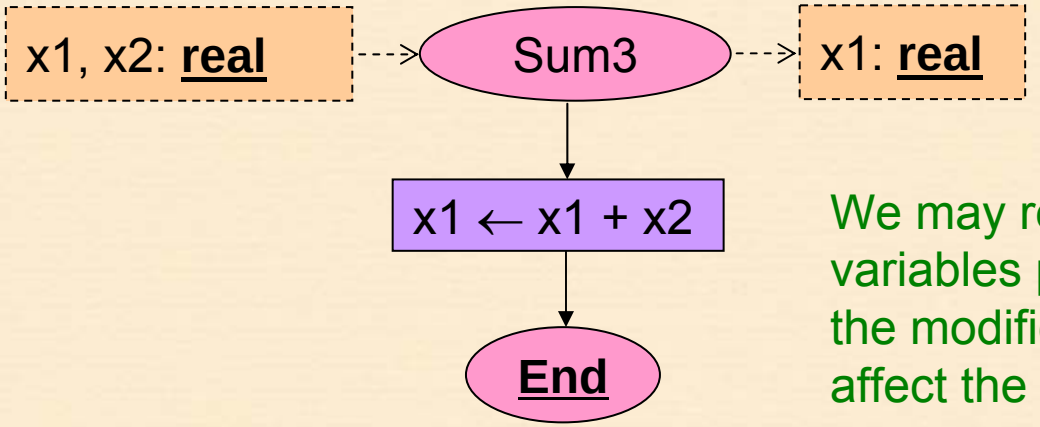
```
Sum2 (5, 7, s)

...
Call Sum2 (5, 7, s)
...
```

It must be a variable to receive the result

2.3 Procedure 2: program and call



2.3 Sum3 procedure: flowchart and VB



We may read the value of the variables passed by reference and the modification carried out on them affect the calling variables

```
Sub Sum3 (ByRef x1 As Double, ByVal x2 As Double)
   x1 = x1 + x2
End Sub
```

It's optional

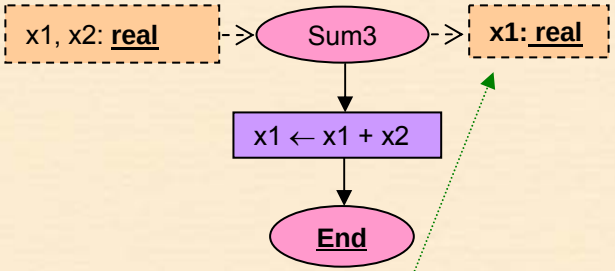
The VB doesn't give us enough information to know if x1 is input or input/output parameter

2.3 Calls to the Sum3 procedure

Call with variables

Sum3 (a, b)

```
...
Call Sum3 (a, b)
...
```



It must be a variable to receive the result

Call with literal constant

Sum3 (a, 7)

```
...
Call Sum3 (a, 7)
...
```

3. Second degree (quadratic) equation

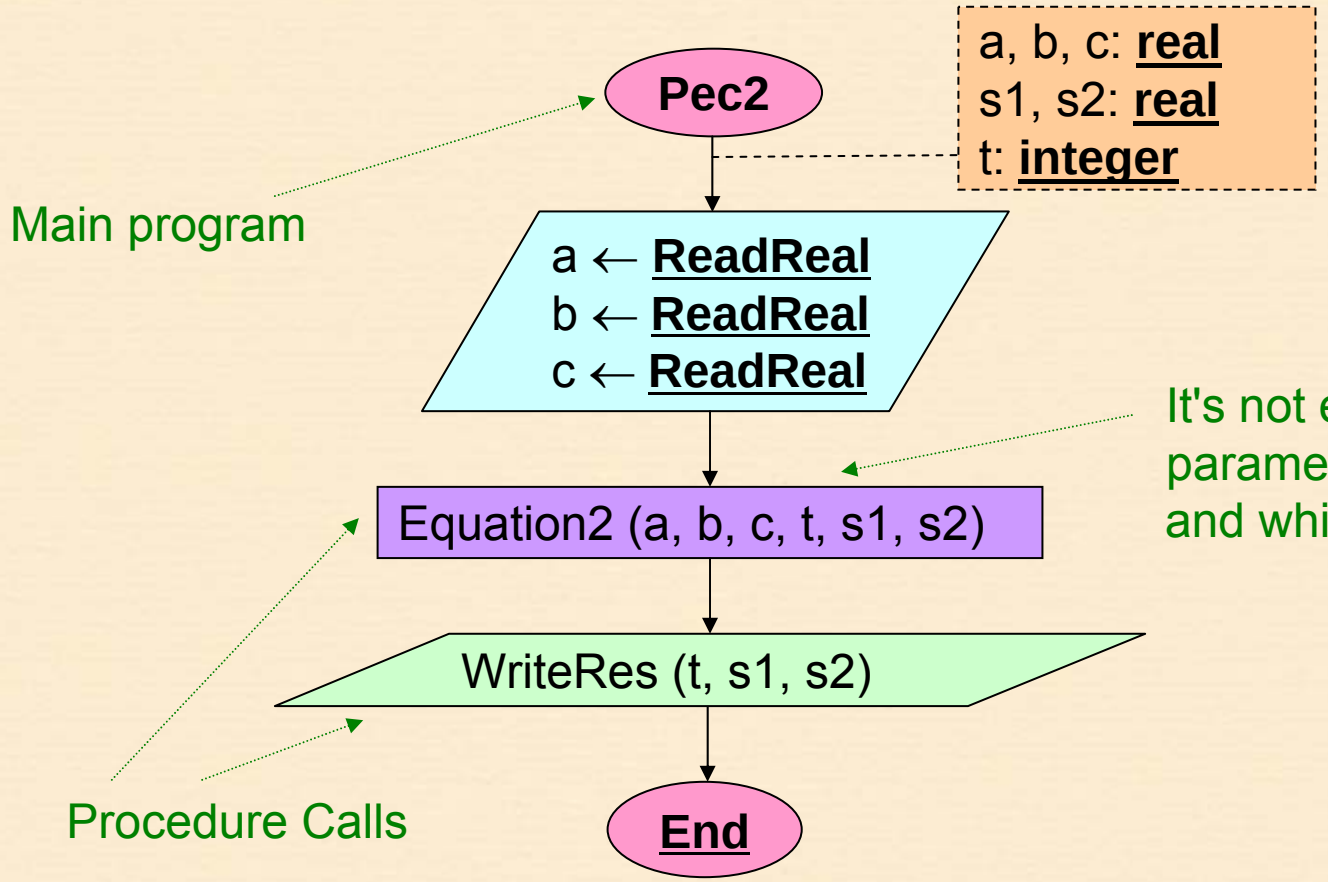
$$ax^2 + bx + c = 0$$

- ***Description***
  - Calculate the roots of a 2nd degree equation
    - Type 0: Not an equation
    - Type 1: Lineal equation: one root
    - Type 2: Two real solutions
    - Type 3: Two imaginary solutions

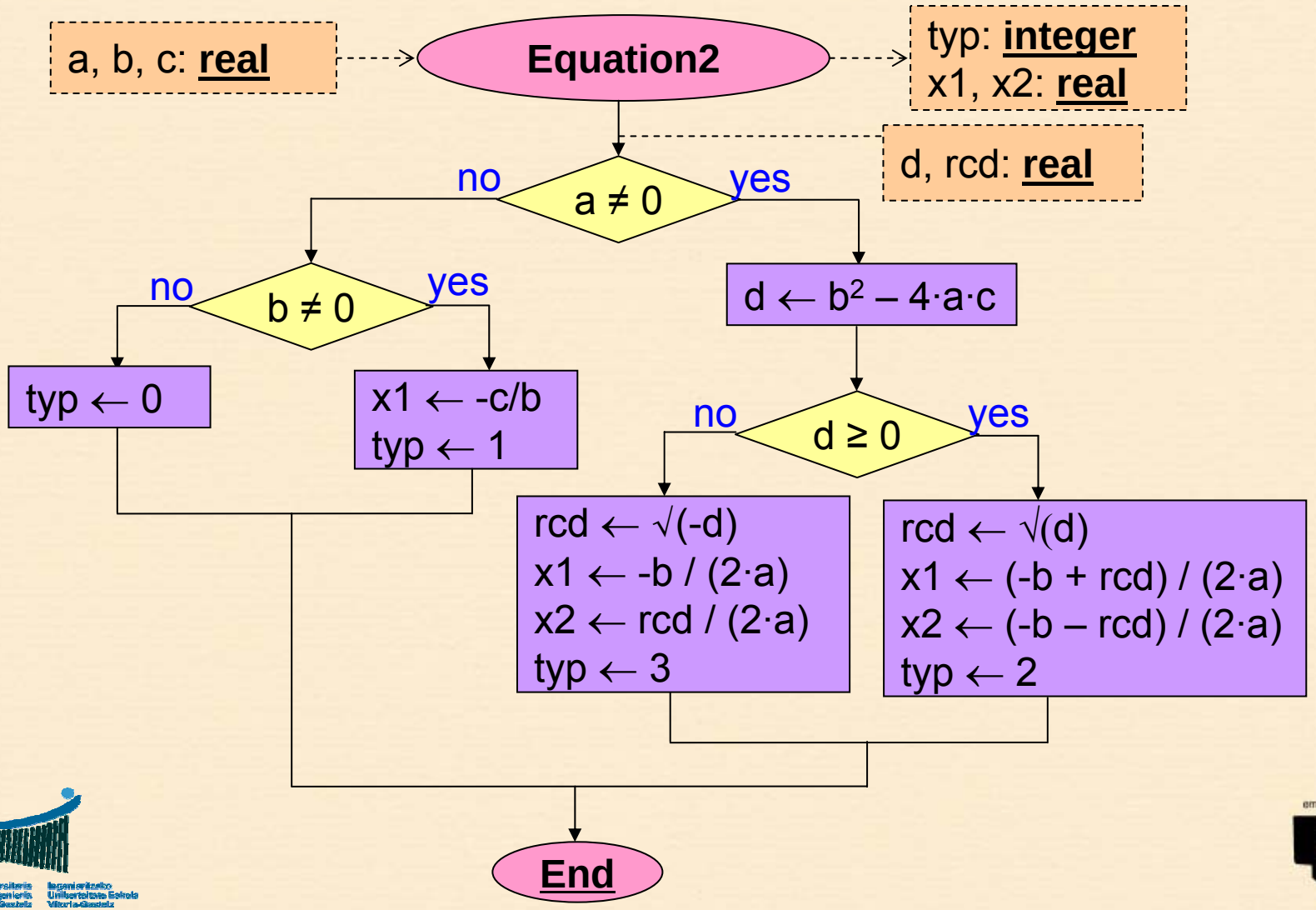
$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

- ***Observations***
  - Passing parameters by reference

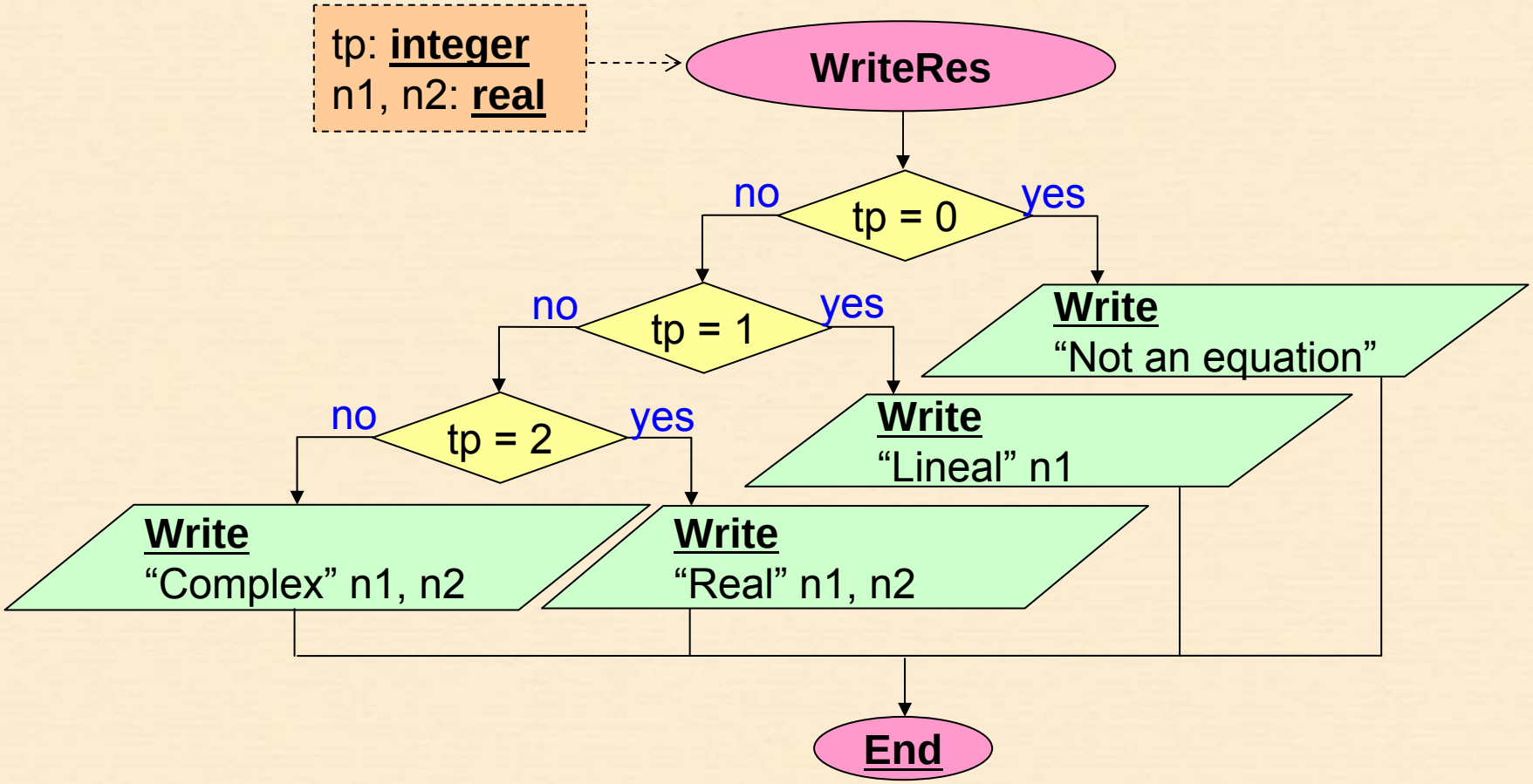
3.1 2nd degree equation flowchart



3.2 Equation2 procedure – flowchart

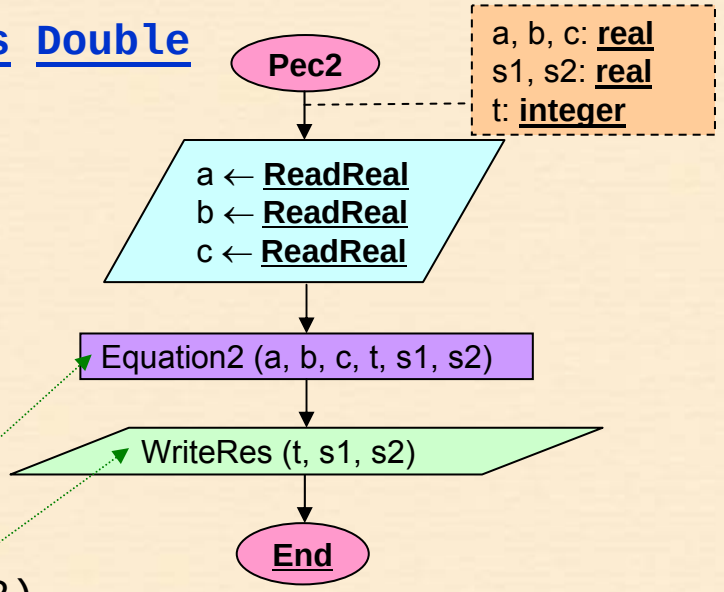


3.3 WriteRes procedure – flowchart



3.4 2nd degree equation VB program

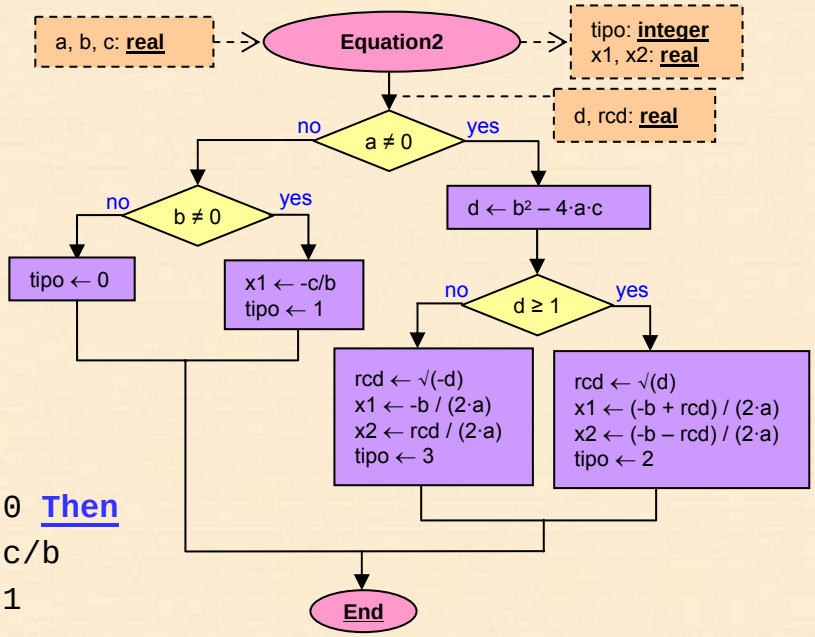
```
Sub Pec2_Click()
  Dim s As String
  Dim a As Double, b As Double, c As Double
  Dim s1 As Double, s2 As Double
  Dim t As Integer
  s = InputBox ("A coefficient")
  a = CDbl (s)
  s = InputBox ("B coefficient")
  b = CDbl (s)
  s = InputBox ("C coefficient")
  c = CDbl (s)
  Call Equation2 (a, b, c, t, s1, s2)
  Call WriteRes (t, s1, s2)
End Sub
```



Procedure calls

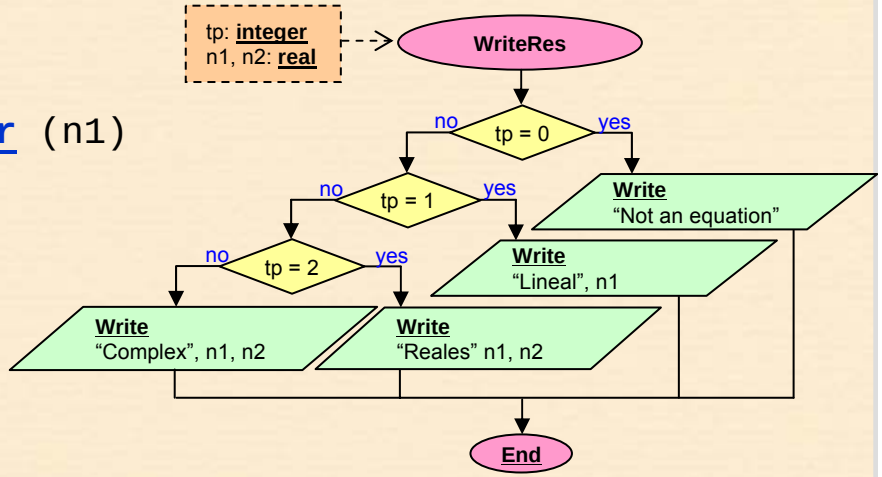
3.5 Equation2 VB procedure

```
Sub Equation2 (ByVal a As Double, ByVal b As Double, ByVal c As Double, _
               ByRef typ As Integer, ByRef x1 As Double, ByRef x2 As Double)
  Dim d As Double, rcd As Double
  If a <> 0 Then
    d = b*b – 4*a*c
    If d >= 0 Then
      rcd = Sqr (d)
      x1 = (-b + rcd)/(2*a)
      x2 = (-b – rcd)/(2*a)
      typ = 2
    Else
      rcd = Sqr (-d)
      x1 = -b /(2*a)
      x2 = rcd / (2*a)
      typ = 3
    End If
  Else
    If b <> 0 Then
      x1 = -c/b
      typ = 1
    Else
      typ = 0
    End If
  End If
End Sub
```

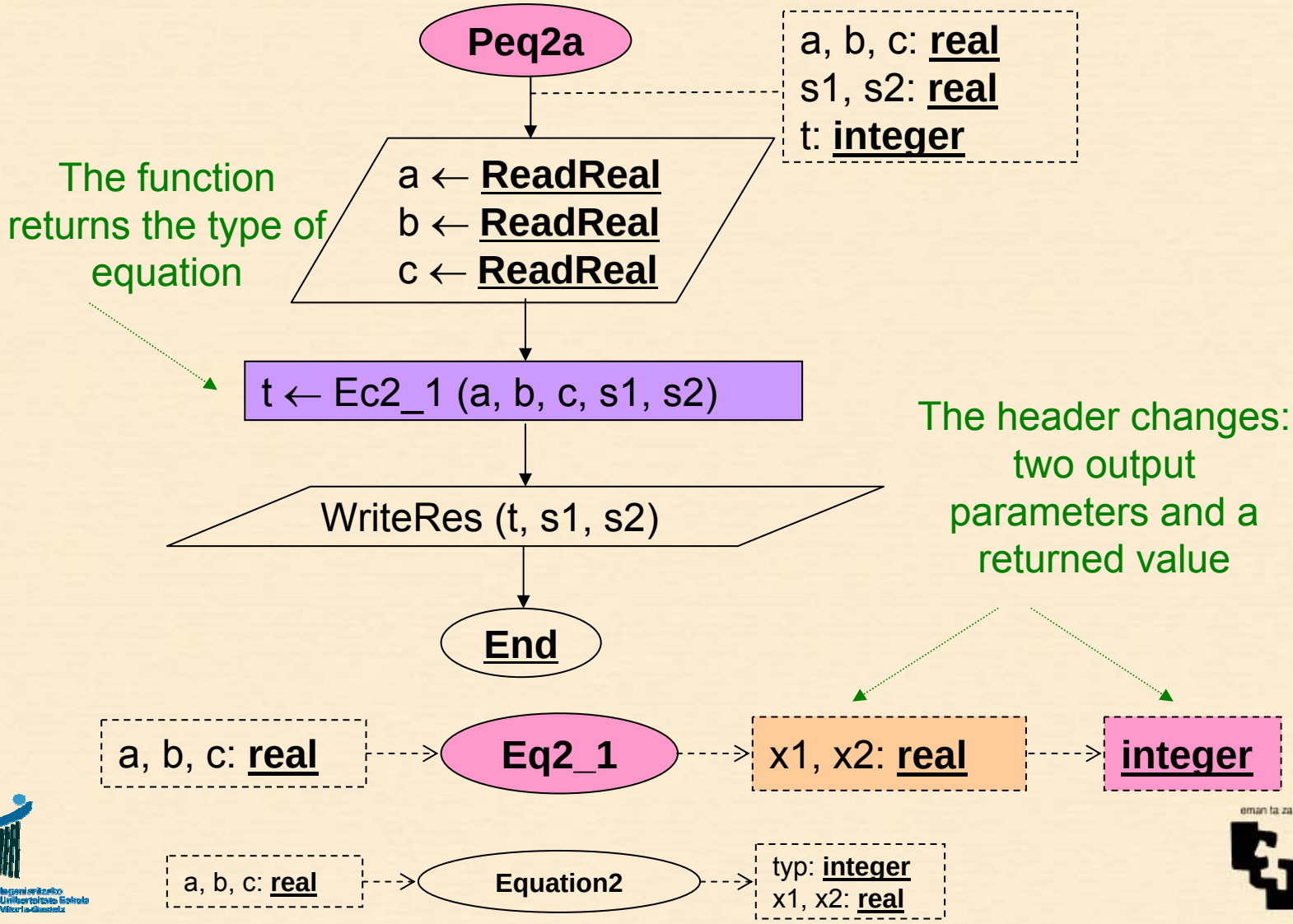


3.6 WriteRes VB procedure

```
Sub WriteRes (ByVal tp As Integer, _
              ByRef n1 As Double, ByRef n2 As Double)
  If tp = 0 Then
    MsgBox "Not an equation"
  ElseIf tp = 1 Then
    MsgBox "Lineal equation. X: " & CStr (n1)
  ElseIf tp = 2 Then
    MsgBox "Real solutions. " & _
           "  x1: " & CStr (n1) & _
           "  x2: " & CStr (n2)
  Else
    MsgBox "Complex solutions. " & _
           " x1: " & CStr (n1) & "+" & CStr (n2) & "i" & _
           " x2: " & CStr (n1) & "-" & CStr (n2) & "i"
  End If
End Sub
```

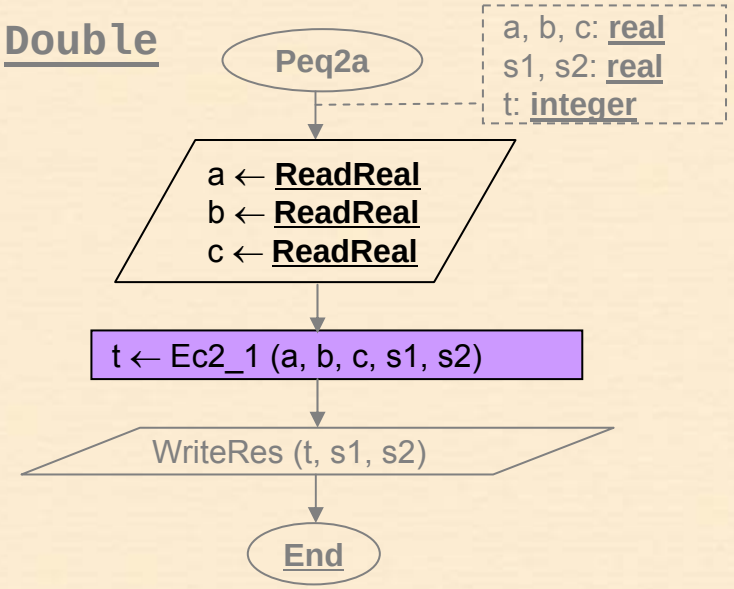


3.7 Alternative program with function



3.8 2nd degree equation VB program

```
Sub Peq2a_Click()
  Dim s As String
  Dim a As Double, b As Double, c As Double
  Dim s1 As Double, s2 As Double
  Dim t As Integer
  s = InputBox ("A coefficient")
  a = CDbl (s)
  s = InputBox ("B coefficient")
  b = CDbl (s)
  s = InputBox ("C coefficient")
  c = CDbl (s)
  t = Ec2_1 (a, b, c, s1, s2)
  Call WriteRes (t, s1, s2)
End Sub
```



3.9 Ec2_1 VB function

```
Function Ec2_1 (ByVal a As Double, ByVal b As Double, ByVal c As Double, _
                ByRef x1 As Double, ByRef x2 As Double) As Integer
  Dim d As Double, rcd As Double
  If a <> 0 Then
    d = b*b – 4*a*c
    If d >= 0 Then
      rcd = Sqr (d)
      x1 = (-b + rcd)/(2*a)
      x2 = (-b – rcd)/(2*a)
      Ec2_1 = 2
    Else
      rcd = Sqr (-d)
      x1 = -b /(2*a)
      x2 = rcd / (2*a)
      Ec2_1 = 3
    End If
  Else
    If b <> 0 Then
      x1 = -c/b
      Ec2_1 = 1
    Else
      Ec2_1 = 0
    End If
  End If
End Function
```

Escuela Universitaria de Ingeniería Vitoria-Gasteiz

Ingeniaritzako Unibertsitate Eskola Vitoria-Gasteiz

eman ta zabal zazu

Universidad del País Vasco

Euskal Herriko Unibertsitatea