Introdução à Computação

Estruturas de Decisão

Conteúdo

- Conceito e aplicação
  - O que são e para que servem as estruturas de decisão.
- Estrutura If ... Then
  - Apresentação das diversas variantes desta estrutura. Expressões lógicas complexas.
- Estruturas de decisão encadeadas
  - Combinação de várias estruturas de forma a tomar decisões mais complexas.

Conceito

- As linguagens de programação têm instruções que permitem tomar decisões com base numa expressão lógica. Em VB temos o IF ... Then
  - Sintaxe (geral)

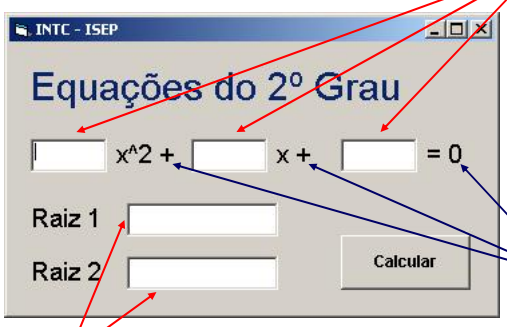
```
If <expressão lógica> then
  <bloco de instruções 1>
[Else
   <bloco de instruções 2>]
End If
```

O código entre parêntesis rectos é opcional, ou seja, não tem necessariamente que existir.

  - Explicação
    - Se a expressão lógica for verdadeira é executado o bloco de instruções 1, caso contrário executa-se o 2.

Exemplo de aplicação

- Pretende-se um programa que resolva equações do 2º grau.
  - Raízes: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$



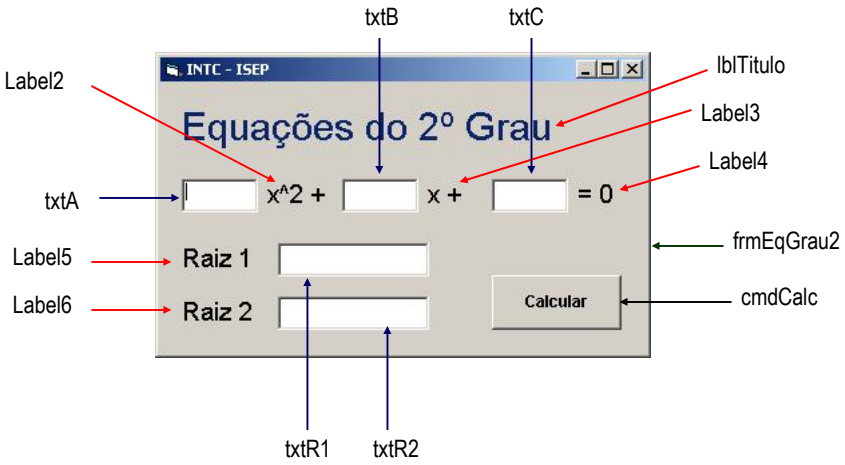
3 *TextBox* para introdução dos coeficientes da equação.

3 *Label* para indicar intuitivamente a função de cada uma das *TextBox*

2 *TextBox* para apresentar os resultados (as duas raízes).

Nomes dos objectos

A *form* do programa já tem o aspecto final, mas falta ainda atribuir nomes aos objectos.



Regras para nomes

- Utiliza-se a notação húngara:

  <prefixo><nome>

  - o prefixo identifica o tipo de objecto.

Tipo	Prefixo	Exemplos
TextBox	`txt`	`txtA, txtR1, txtNumAlunos`
Label	`lbl`	`lblNumAlunos, lblA`
CommandButton	`cmd`	`cmdCalc, cmdSair`
Form	`frm`	`frmEqGrau2`
ListBox	`lst`	`lstAlunos`

  - o nome está relacionado com a função do objecto no programa e deve permitir a sua fácil identificação.

O código

O código do programa é o seguinte:

```
Option Explicit

Private Sub cmdCalc_Click()

    Dim a As Single, b As Single, c As Single
    Dim delta As Single

    a = txtA.Text
    b = txtB.Text
    c = txtC.Text

    delta = Sqr(b ^ 2 - 4 * a * c)
    txtR1.Text = (-b + delta) / 2 / a
    txtR2.Text = (-b - delta) / (2 * a)
End Sub
```

Instrução para o interpretador do VB

Definição das variáveis

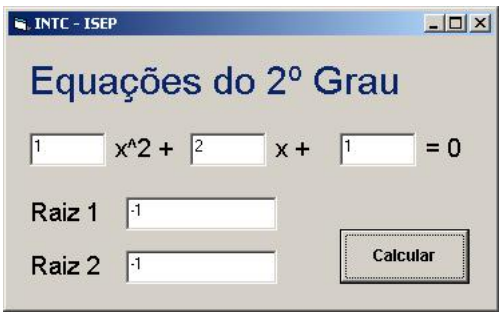
Leitura dos valores contidos nas três caixas de texto. Não é verificado se os valores são válidos.

`Sqr()` é uma função do VB que calcula a raiz quadrada de um número positivo

A variável delta permite economizar alguns cálculos, não sendo necessário repetir o cálculo da raiz quadrada para a segunda raiz.

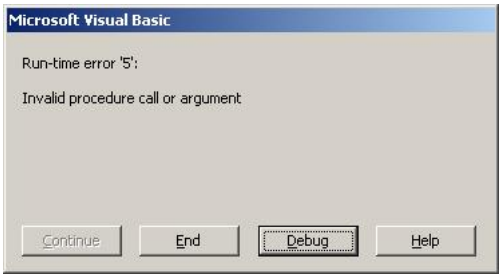
Testar

Uma vez terminado o programa, vamos efectuar testes. Para tal, introduzem-se valores conhecidos: 1, 2, 1 (raiz dupla –1).



Testar II

Vamos testar com outros valores: 1, 0, 1 (raízes complexas +i e -i).



Deu erro e o programa parou! A função **`Sqr()`** não aceita argumentos negativos!

Solução

Não podemos deixar que função `Sqr()` receba argumentos negativos.

```
Option Explicit
Private Sub cmdCalc_Click()
    Dim a As Single, b As Single, c As Single
    Dim delta As Single

    txtR1.Text = ""
    txtR2.Text = ""
    a = txtA.Text
    b = txtB.Text
    c = txtC.Text

    delta = b ^ 2 - 4 * a * c

    If delta >= 0 Then
        txtR1.Text = (-b + Sqr(delta)) / 2 / a
        txtR2.Text = (-b - Sqr(delta)) / (2 * a)
    Else
        MsgBox "Raízes complexas!"
    End If
End Sub
```

Limpa o conteúdo inicial das TextBox com as raízes, de forma a que os resultados anteriores não apareçam.

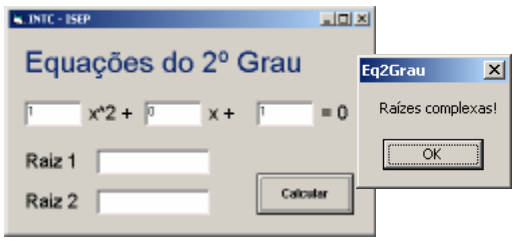
`delta` representa agora o interior da raiz

Se `delta` for maior ou igual a zero então é possível calcular a raiz quadrada. Existem raízes reais para a equação.

Mensagem a indicar que as raízes são complexas.

Testando novamente

Utilizamos os valores: 1, 0, 1 (raízes complexas +i e -i).



O programa não dá erro, informando antes o utilizador da natureza complexa das raízes!

Desenvolvimento:

O programa ainda não está perfeito, podendo ainda dar erro devido aos valores introduzidos pelo utilizador. Tente descobrir e eliminar essa possibilidade (dica: as divisões são sempre uma fonte de problemas!). Tente também apresentar as raízes complexas da equação.

Decisões complexas

- Problema:
  - Pretende-se elaborar um programa que determine qual o maior de três números dados pelo utilizador.
- Solução
  - Há duas abordagens possíveis para resolver este problema:
    - utilizar três estruturas de decisão independentes
      - Cada estrutura avalia uma expressão que, por si só, permite definir o resultado.
    - utilizar duas estruturas de decisão encadeadas
      - As estruturas estão dependentes dos resultados das estruturas anteriores. A expressão avaliada numa dada estrutura não define completamente o resultado final.

Decisões independentes

- Cada `If ... then` determina se um número é o maior dos três:

```
Private Sub cmdCalc_Click()
    Dim a As Integer, b As Integer, c As Integer
    Dim max As Integer

    txtMax.Text = ""
    a = txtA.Text
    b = txtB.Text
    c = txtC.Text

    If a > b And a > c Then
        max = a
    End If
    If b > a And b > c Then
        max = b
    End If
    If c > a And c > b Then
        max = c
    End If
    txtMax.Text = max
End Sub
```

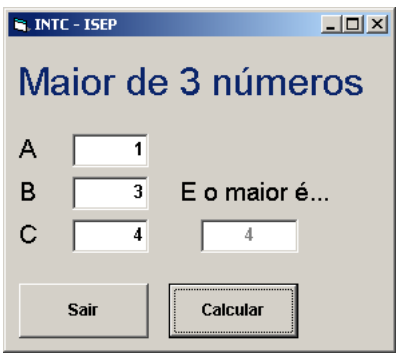
Repare-se que `max`, a variável que vai ter o maior dos números, é do mesmo tipo que as variáveis que contêm os valores lidos.

Se `b` for maior do que `a` e do que `c`, então é certamente o maior número.

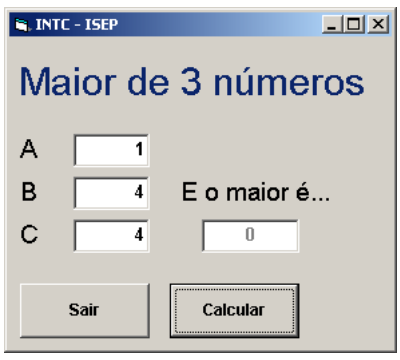
Apresenta-se o maior numa TextBox.

Teste do programa

E o resultado está correcto!



Ou talvez não!...



Esquecemos-nos que poderia haver números iguais!

Decisões independentes II

- Cada `If ... then` determina se um número é o maior dos três:

```
Private Sub cmdCalc_Click()
    Dim a As Integer, b As Integer, c As Integer
    Dim max As Integer

    txtMax.Text = ""
    a = txtA.Text
    b = txtB.Text
    c = txtC.Text

    If a >= b And a >= c Then
        max = a
    End If
    If b >= a And b >= c Then
        max = b
    End If
    If c >= a And c >= b Then
        max = c
    End If
    txtMax.Text = max
End Sub
```

Se a for maior ou igual a b e maior ou igual a c, então é certamente o maior número.

Decisões encadeadas

- Cada `If...then` avalia só parte da condição completa:

```
Private Sub cmdCalc_Click()
    Dim a As Integer, b As Integer, c As Integer
    Dim max As Integer

    txtMax.Text = ""

    a = txtA.Text
    b = txtB.Text
    c = txtC.Text

    If a >= b And a >= c Then
        max = a
    Else
        If b > c Then
            max = b
        Else
            max = c
        End If
    End If
    txtMax.Text = max
End Sub
```

Temos duas estruturas de decisão encadeadas, a segunda dentro da "componente falsa" da primeira.

A primeira condição define completamente se a é o maior dos três...

Mas a segunda só verifica se b é maior do que c. Note-se que a não é garantidamente o maior, senão a primeira condição era verdadeira.

Se nenhum dos anteriores era o maior, então é c.

Decisões encadeadas II

- Também podia ser usada a estrutura **`If...ElseIf`**:

```
Private Sub cmdCalc_Click()
    Dim a As Integer, b As Integer, c As Integer
    Dim max As Integer

    txtMax.Text = ""

    a = txtA.Text
    b = txtB.Text
    c = txtC.Text

    If a >= b And a >= c Then
        max = a
    ElseIf b > c Then
        max = b
    Else
        max = c
    End If
    txtMax.Text = max
End Sub
```

Em vez de construir uma cascata de Ifs, esta estrutura permite aumentar a legibilidade do programa quando as decisões são mutuamente exclusivas. O resultado prático é exactamente o mesmo do exemplo anterior.

Atenção, que podem ser usados vários `ElseIf`, mas só pode haver um `Else`!

Conclusão

- Nesta aula introduzimos as estruturas de decisão, que permitem aos programas tomar decisões e simular até alguma "inteligência".
- Apresentamos duas formulações:
  - **`If … Then … [Else …] End If`**
  - **`If … Then … ElseIf …[Else …] End If`**
- Estes conceitos vão ser explorados nas aulas práticas, resolvendo exercícios.