Introdução à Computação

Módulo 5

Vectores e Matrizes

Conteúdo

- Conceito e aplicação
  - O que são e para que servem as variáveis indexadas unidimensionais (vectores) e bidimensionais (matrizes).
- Vectores
  - Sintaxe e exemplos de aplicação.
- Matrizes
  - Sintaxe e exemplos de aplicação.
- Passagem de vectores e matrizes como parâmetros de rotinas.

Variáveis Indexadas

- O que são variáveis indexadas?
  - Uma grande parte das linguagens de programação permitem agrupar variáveis de um mesmo tipo num único bloco, em que cada variável pode ser manipulada directamente indicando a sua posição (índice) dentro do bloco.
  - As variáveis indexadas permitem então manipular blocos de informação de tamanho arbitrário (ainda que normalmente limitado) de forma expedita.
  - Nesta disciplina consideramos dois tipos de variáveis indexadas:
    - Vectores – variáveis indexadas unidimensionais, em que a posição de um elemento é dada por um só índice.
    - Matrizes – variáveis indexadas bidimensionais, em que a posição de um elemento é dada por dois índices, referentes à linha e coluna.

Vectores - Sintaxe

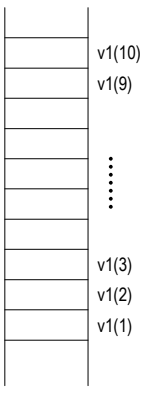
- Sintaxe
  - Geral

```
Dim <nome>(<dimensão>) as <tipo de dados>
```

  - Exemplo

```
Const NumAlunos = 60
Dim v(50) as Single, v1(10) as Integer
Dim turma(NumAlunos) as Integer
```

NumAlunos é uma constante que representa o valor 60, logo é o mesmo que escrever o literal 60.

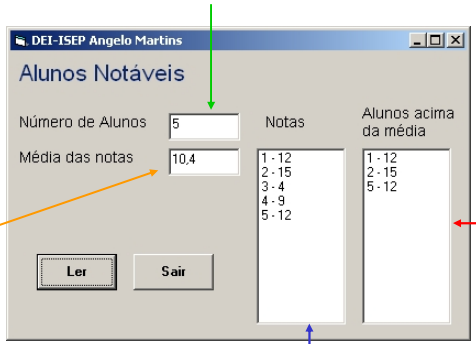


Vectores – Exemplo

- Elabore um programa que leia as notas de uma turma e que apresente numa ListBox as notas dos alunos que estejam acima da media. Uma turma pode ter no máximo 60 alunos.
  - Para identificar as notas que se encontram acima da média, é necessário calcular primeiro a média. Isso obrigaria a ler as notas duas vezes, se não houvesse a possibilidade de guardar as notas num vector.
  - No enunciado do problema nada é dito quanto ao número de alunos a ler, só que há um limite de 60. Assim, decidimos que o utilizador vai ter de introduzir o número de alunos numa TextBox.

Vectores - Form

txtNA - *TextBox* para introdução do número de alunos.



txtMed - *TextBox* para apresentação da média.

lstQH - *ListBox* com os alunos acima da média.

lstNotas - *ListBox* para apresentação das notas de toda a turma.

Resolução I

```
Option Explicit
Private Sub cmdSair_Click()
    End
End Sub

Private Sub cmdLer_Click()
    Dim notas(60) As Integer, soma As Long
    Dim x As Integer, med As Single, na As Integer

    lstNotas.Clear
    lstQH.Clear
    txtMed.Text = ""

    na = Val(txtNA.Text)
    If na < 1 Or na > 60 Then
        MsgBox "Erro: N° de alunos inválido!", vbCritical
        Exit Sub
    End If
```

Vector com capacidade para 60 inteiros.

Limpa os valores de corridas anteriores.

É fundamental verificar que o número de alunos é positivo e não ultrapassa a capacidade do vector.

Resolução II

(continuação)

```
    For x = 1 To na
        Do
            notas(x) = Val(InputBox("Nota do aluno " & x))
        Loop While notas(x) < 0 Or notas(x) > 20
        soma = soma + notas(x)
    Next
    med = soma / na

    For x = 1 To na
        lstNotas.AddItem x & " - " & notas(x)

        If notas(x) > med Then
            lstQH.AddItem x & " - " & notas(x)
        End If
    Next
    txtMed.Text = med
End Sub
```

Leitura do valor para a posição **x** do vector. O valor de **x** vai variar entre 1 e o valor de **na**.

Se a nota actual for superior à média, então adiciona-a à *ListBox*.

Matrizes - Sintaxe

- Sintaxe
  - Geral

```
Dim <nome>(<nº de linhas>, <nº de colunas>) as <tipo de dados>
```

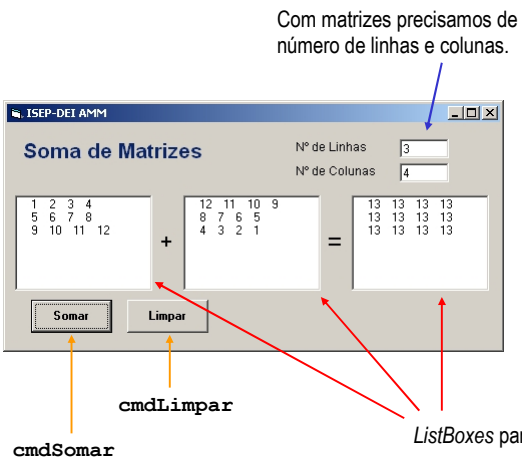
  - Exemplo

```
Dim turma( 30 , 6) as Integer
```

turma é uma matriz com 30 linhas e 6 colunas, ou seja, permite guardar as notas de 30 alunos a 6 disciplinas.

Matrizes - Exemplo

Elabore um programa que some duas matrizes.



Resolução I

```
Option Explicit

Const Linhas = 10
Const Colunas = 10

Private Sub cmdLimpar_Click()
    lstM1.Clear
    lstM2.Clear
    lstMS.Clear
End Sub

Private Sub cmdCalc_Click()
    Dim m1(Linhas, Colunas) As Integer, m2(Linhas, Colunas) As Integer
    Dim ms(Linhas, Colunas) As Integer, linha As String
    Dim nl As Integer, nc As Integer, i As Integer, j As Integer

    cmdLimpar_Click     ' equivale a carregar no botão cmdLimpar

    nl = Val(txtNL.Text)
    nc = Val(txtNC.Text)
    If nc < 1 Or nc > Colunas Or nl < 1 Or nl > Linhas Then
        MsgBox "Valores errados!"
        Exit Sub
    End If
```

Definem-se duas constantes com as dimensões máximas das matrizes

Utilizar as duas constantes ou o valor 10 é exactamente o mesmo.

É preciso validar õ número de linhas e colunas.

Resolução II

```
    For i = 1 To nl          ' leitura da matriz 1
        linha = ""
        For j = 1 To nc
            m1(i, j) = Val(InputBox("M1 [ " & i & ", " & j & "] = "))
            linha = linha & "    " & m1(i, j)
        Next
        lstM1.AddItem linha
    Next
    For i = 1 To nl          ' leitura da matriz 2
        linha = ""
        For j = 1 To nc
            m2(i, j) = Val(InputBox("M2 [ " & i & ", " & j & "] = "))
            linha = linha & "    " & m2(i, j)
        Next
        lstM2.AddItem linha
    Next
    For i = 1 To nl
        linha = ""
        For j = 1 To nc
            ms(i, j) = m2(i, j) + m1(i, j)
            linha = linha & "    " & ms(i, j)
        Next
        lstMS.AddItem linha
    Next
End Sub
```

(continuação)

Leitura de um elemento da matriz.

No início uma linha tem que estar vazia.

Acrescenta um elemento da matriz à linha.

Escreve uma linha na *ListBox*.

Utilização de Subrotinas e Funções

Elabore uma função que retorne a média dos elementos de uma matriz de números inteiros.

As matrizes e os vectores são sempre passados por referência.

```
Function MedMat( ByRef m() as Integer , ByVal nl as Integer, _
                 ByVal nc as Integer) as Single

    Dim x As Integer, y As Integer, soma As Long

    For x = 1 to nl
       For y = 1 to nc
           soma = soma + m(x,y)
       Next
    Next
    MedMat = soma / (nl * nc)
End Sub
```

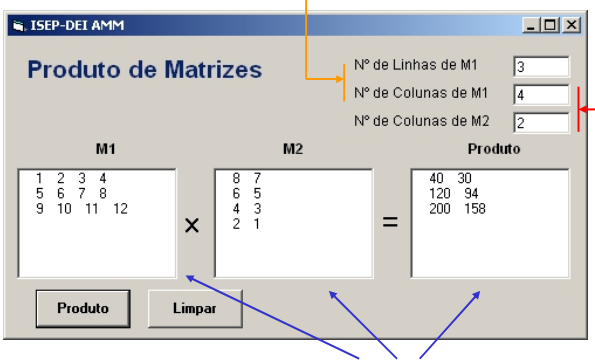
Juntamente com uma matriz é necessário passar parâmetros com as dimensões da matriz realmente utilizadas.

Número de elementos da matriz.

Exemplo da Aplicação

Elabore um programa que calcule o produto de duas matrizes.

Dimensões da matriz M1



Dimensões da matriz M2

ListBoxes para representar as matrizes.

Resolução I

```
Option Explicit

Const Linhas = 10
Const Colunas = 10

Private Sub cmdLimpar_Click()
    lstM1.Clear
    lstM2.Clear
    lstMP.Clear
End Sub

Sub LerMat(mat() As Integer, ByVal nl As Integer, _
           ByVal nc As Integer, ByVal stNome As String)

    Dim x As Integer, y As Integer

    For x = 1 To nl
        For y = 1 To nc
            mat(x, y) = Val(InputBox(stNome & "[ " & x & ", " & _
                        y & "] = "))
        Next
    Next
End Sub
```

Dimensões máximas das matrizes.

Nome da matriz, para a subrotina poder ser usada para a leitura de diferentes matrizes.

Subrotina de leitura de uma matriz.

Resolução II

Dimensões de **m1**.

Dimensões de **m2**.

```
Sub ProdutoMat(m1() As Integer, m2() As Integer, p() As Integer, _
           ByVal nl1 As Integer, ByVal nc1 As Integer, ByVal nc2 As Integer)

    Dim x As Integer, y As Integer, w As Integer

    For x = 1 To nl1
        For y = 1 To nc2

            p(x, y) = 0
            For w = 1 To nc1
                p(x, y) = p(x, y) + m1(x, w) * m2(w, y)
            Next

        Next
    Next
End Sub
```

Dimensões de **p**.

Calcula o produto de uma linha de **m1** por uma coluna de **m2** e guarda-o na matriz **p**. Este cálculo é feito para cada elemento de **p**.

Resolução III

```
Sub MostrarMatriz(m() As Integer, ByVal nl As Integer, _
            ByVal nc As Integer, ls As ListBox)

    Dim i As Integer, j As Integer, linha As String
    For i = 1 To nl
        linha = ""
        For j = 1 To nc
            linha = linha & "    " & m(i, j)
        Next
        ls.AddItem linha
    Next
End Sub

Private Sub cmdProd_Click()
    Dim m1(Linhas, Colunas) As Integer, m2(Linhas, Colunas) As Integer
    Dim mp(Linhas, Colunas) As Integer, linha As String
    Dim nl1 As Integer, nc1 As Integer, nc2 As Integer
    Dim i As Integer, j As Integer

    cmdLimpar_Click     ' equivale a carregar no botão cmdLimpar
```

A *Listbox* é passada como parâmetro para a subrotina poder ser usada com todas as matrizes.

Utilização normal de uma *ListBox*.

Resolução IV

(continuação)

Leitura e validação das dimensões.

```
    nl1 = Val(txtNLM1.Text)
    nc1 = Val(txtNCM1.Text)
    nc2 = Val(txtNCM2.Text)
    If nc1 < 1 Or nc1 > Colunas Or nl1 < 1 Or nl1 > Linhas Or _
       nc2 < 1 Or nc2 > Colunas Then
        MsgBox "Valores errados!"
        Exit Sub
    End If

    LerMat m1, nl1, nc1, "Matriz 1"
    MostrarMatriz m1, nl1, nc1, lstM1
    LerMat m2, nc1, nc2, "Matriz 2"
    MostrarMatriz m2, nc1, nc2, lstM2
    ProdutoMat m1, m2, mp, nl1, nc1, nc2
    MostrarMatriz mp, nl1, nc2, lstMP
End Sub
```

Visualização das matrizes.

Leituras das matrizes.

Cálculo do produto.

Conclusão

- Há grandes vantagens na utilização de subrotinas e funções com matrizes e vectores:
  - As rotinas genéricas podem ser reutilizadas.
    - Exemplo: **`LerMat()`** e **`MostrarMatriz()`**
  - O código no botão fica muito mais simples e legível.
  - Cometem-se menos erros a programar e as alterações são mais localizadas.
- Também há desvantagens:
  - Código menos eficiente.
  - Exige mais conhecimentos do programador.