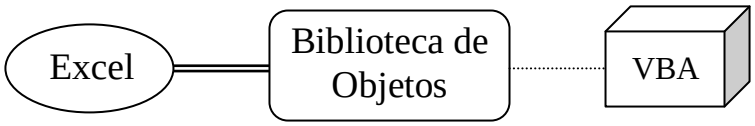
Visual Basic Application para Excel

Prof. Luiz Bianchi
Universidade Regional de Blumenau

Sumário

Introdução

A versão de Visual Basic para Aplicativos é um ambiente completo de desenvolvimento, consistente com a versão de plataforma única do Visual Basic e compartilhada por todos os aplicativos do Microsoft Office.
O Visual Basic interpreta um conjunto especial de comandos denominado biblioteca de objetos do Excel. O Visual Basic que vem com o Excel não é a única linguagem que poderá comunicar-se com a biblioteca de objetos. Qualquer linguagem que ofereça suporte à automação poderá controlar o Excel.



Conceitos

Alguns componentes essenciais da linguagem VBA para Excel são, a seguir, destacados e conceituados.

Objetos

Um objeto é um tipo especial de variável que contém dados e códigos e representa um elemento específico no Excel. O Visual Basic suporta um conjunto de objetos que correspondem diretamente aos elementos do Microsoft Excel.

Por exemplo, o objeto **Workbook** representa uma pasta de trabalho, o objeto **Worksheet** representa uma planilha e o objeto **Range** representa um intervalo de células.

Uma **pasta de trabalho**, no Microsoft Excel, corresponde a um arquivo que pode conter diversas planilhas e folhas de gráficos ou planilhas de gráficos.

Propriedades e métodos

Para realizar uma tarefa o Visual Basic retorna um objeto que representa o elemento apropriado do Excel e depois o manipula usando as **propriedades** e **métodos** daquele objeto.

As **propriedades** são características ou atributos de um objeto e os **métodos** são ações que os objetos podem executar.

Procedimento

Um procedimento é uma unidade de código localizada entre instruções **Sub** e **End Sub** ou entre instruções **Function** e **End Function** que realiza uma tarefa.
Um procedimento desempenha uma tarefa específica. Um procedimento Function pode retornar valor, ao passo que um procedimento Sub não retorna valor.

Para uma visão geral da estrutura de um procedimento **Sub,** segue um exemplo com breves comentários que explicam cada linha:

```
Sub ObterNome()
' Declara um procedimento Sub que não utiliza argumentos
    Dim resposta As String
    ' Declara uma variável de seqüência de caracteres de nome resposta
    resposta = InputBox(Prompt:="Qual é o seu nome?")
    ' Atribui o valor de retorno da função InputBox à resposta
    If resposta = Empty Then
    ' Instrução condicional If...Then...Else
        MsgBox Prompt:="Você não digitou um nome."
        ' Chama a função MsgBox
    Else
        MsgBox Prompt:="O seu nome é " & resposta
        ' Função MsgBox concatena com a variável resposta
    End If
    ' Encerra a instrução If...Then...Else
End Sub
' Encerra o procedimento Sub
```

Nota: As frases iniciadas por um apóstrofo (') são comentários admitidos na codificação do programa VBA, os quais não são analisados pelo compilador.

Módulo

Um módulo é um conjunto de procedimentos que realiza tarefas específicas.
Por exemplo, procedimentos que executam várias tarefas contábeis podem ser agrupados em um módulo.

Fundamentos

Neste tópico serão tratados os comandos básicos da linguagem VBA para excel.

Declaração de variáveis

Uma variável é uma área na memória, identificada por um nome, onde pode ser armazenado um valor e alterado a qualquer momento.

A variável pode ser declarada de modo implícito pelo VBA no momento em que ela for referenciada numa instrução. No entanto, o programa poderá tornar-se mais eficiente se as variáveis forem declaradas de modo explícito pelo usuário. A declaração explícita de todas as variáveis reduz a incidência de erros de conflitos de nomenclatura e de digitação.

Para impedir que o VBA faça declarações implícitas, deve-se inserir a instrução **Option explicit** em um módulo antes de todos os procedimentos.

Uma variável pode ser declarada, usando as seguintes palavras-chave para definir seu escopo:

Dim ou **Static**	(no procedimento)
Dim ou **Private**	(no módulo)
Public	(no módulo)

Dim – O valor da variável é retido apenas enquanto o procedimento no qual ela foi declarada estiver em execução.
Static – a variável preserva o valor entre as chamadas ao procedimento.
Private – o valor fica disponível a todos os procedimentos dentro do módulo onde a variável foi declarada.
Public – a variável pode ser acessada pelos procedimentos de vários módulos de uma pasta de trabalho.

Exemplos:
```
Dim area, valor
Static acumula
Private inteiro
```

Constantes

Uma variável declarada por meio do qualificador **const** significa que seu conteúdo não poderá ser alterado em todo programa. A constante deve ser inicializada, isto é, no momento de sua declaração deverá ser atribuído um valor a ela.

Exemplo: `Const pi = 3.1416`

Fontes de constantes:

Constantes predefinidas – são fornecidas pelos aplicativos (Excel, Access, Project, etc).
Exemplo de uma constante: `xlAbsolute`

Constantes simbólicas ou definidas pelo usuário – são declaradas através da instrução **Const**.

Exemplos:

```
Const pi = 3.1416
Const Pi2 = Pi * 2
```

Pode ser especificado o escopo de uma constante, como segue:

`Private Const Pi = 3.14159` Fica disponível a todos os procedimentos dentro de um dado módulo. Deve ser declarada a nível de módulo.

`Public Const max = 1024` Permanece disponível a todos os módulos. Deve ser declarada a nível de módulo.

`Const idade = 29` Disponível apenas dentro do procedimento onde foi declarada.

Tipos de dados

O tipo de uma variável determina a quantidade de memória que ela ocupará, em bytes, e o modo de armazenamento. O VBA opera com os seguintes tipos básicos:

Nome	Tamanho	Intervalo
Integer	2 bytes	-32768 a 32767
Long	4 bytes	-2.147.483.648 a 2.147.483.467
Single	4 bytes	$-3{,}4 \times 10^{38}$ a $3{,}4 \times 10^{38}$
Double	8 bytes	$1{,}7 \times 10^{308}$ a $1{,}7 \times 10^{308}$
Currency	8 bytes	-9223372036854,5808 a 9223372036854,5807
String	1 byte por caractere	0 a aproximadamente 65.500
Boolean	2 bytes	Verdadeiro ou Falso
Date	8 bytes	01/01/100 a 31/12/9999
Object	4 bytes	Qualquer referência a objeto
Variant	16 bytes + 1 byte para cada caractere	Válido para qualquer tipo de dados.

Numérico

Uma variável que conterá número inteiro pode ser declarada como **Integer** ou **Long**.

Exemplos:
```
Dim contador As Integer
Private tamMemoria As Long
```

Uma variável que conterá números fracionários, pode ser declarada com o tipo de dado **Single**, **Double** ou **Currency**.

Exemplos:
```
Public lado1 As Single
Private área As Double
Dim custoProd As Currency
```

String

Uma variável que conterá um conjunto de caracteres alfanuméricos pode ser declarada com o tipo de dados **String**.

Exemplos:
```
Dim descrProd As String
Dim nomeFunc As String
```

Boolean

Uma variável que contém valor lógico (verdadeiro ou falso) pode ser declarada com o tipo de dados Boolean. O valor padrão é **False**.

Exemplo:
```
Dim limExcedido As Boolean
```

Date

Uma variável que contém valores de data e hora deve ser declarada com o tipo de dados **Date**.

Exemplo:
```
data As Date
```

Atribuição de data e hora em literais Date:

```
data = #10/2/2001 11:20 AM#
```

Pode-se efetuar cálculos com valores de data e hora. Para adicionar, por exemplo, 20 dias soma-se 20 à variável e para subtrair 1 hora, diminui-se 1/24 da variável.

Object

Uma variável que contém uma referência a um objeto do MS Excel pode ser declarada com tipo de dados **Object**. Para atribuir um objeto a uma variável-objeto, deve-se usar a instrução **Set**.

Exemplos:
```
Dim plan1 As object
Set plan1 = Worksheets(1)
```

Variant

Uma variável **Variant** permite o armazenamento de qualquer tipo de dado.

Exemplo:
```
Dim codMarca      'Variant por padrão
```

Operador de atribuição

O operador de atribuição é representado por = (sinal de igualdade). Atribui a expressão à direita do sinal de igualdade à variável a sua esquerda.
Exemplo:

```
x = 5
```

é atribuído o valor 5 à variável de nome x.

Caixa de entrada – função InputBox

A função **InputBox** apresenta uma caixa de diálogo para que o usuário possa introduzir o dado de entrada.
Ela exibe um aviso em uma caixa de diálogo, aguarda até que o usuário insira texto ou clique em um botão e retorna o conteúdo da caixa de texto.

Sintaxe:

InputBox(*prompt*[, *title*] [, *default*] [, *xpos*] [, *ypos*] [, *helpfile*, *context*]**)**

onde,

prompt	argumento obrigatório e representa a mensagem que será exibida na caixa de diálogo;
title	opcional; texto a ser exibido na barra de título da caixa de diálogo;
default	opcional; dado padrão de entrada a ser exibido na caixa de texto;
xpos e ypos	opcionais; especificam as coordenadas para posicionamento da caixa de diálogo na tela;
helpfile e context	opcionais; identifica o arquivo de ajuda do usuário e o número de contexto atribuído ao tópico da ajuda.

Observação: Para omitir alguns argumentos posicionais, deve-se incluir o delimitador de vírgula correspondente.

Exemplo: `nome = InputBox("Qual é o seu nome?", "Entrada de nomes")`

Essa instrução ao ser executada, surge a seguinte caixa de diálogo na tela:

Neste exemplo, o nome informado pelo usuário na caixa de diálogo é atribuido a variável nome.

Caixa de saída – função MsgBox

Mostra uma caixa de diálogo contendo o botão OK e o valor do dado de saída.

Sintaxe:

MsgBox(*prompt*[, *buttons*] [, *title*] [, *helpfile*, *context*]**)**

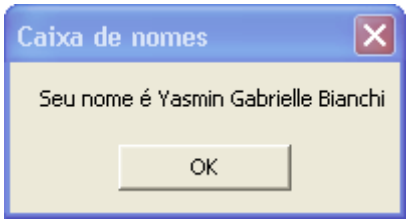
onde,

prompt	argumento obrigatório; mensagem que será exibida na caixa de diálogo;
buttons	opcional; especifica o tipo de botão a ser exibido na caixa de diálogo;
title	opcional; texto a ser exibido na barra de título da caixa de diálogo;
helpfile e context	opcionais; identifica o arquivo de ajuda do usuário e o número de contexto atribuído ao tópico da ajuda.

Observação: Para omitir alguns argumentos posicionais, deve-se incluir o delimitador de vírgula correspondente.

Exemplo: `MsgBox "Seu nome é " & nome, , "Caixa de nomes"`

Nome é a variável que contém o dado ou o nome da pessoa a ser exibido na caixa de saída juntamente com o primeiro texto digitado entre aspas. O segundo texto refere-se ao título da caixa de diálogo. Ao lado, vê-se a figura da caixa de diálogo de saída.



Comentários

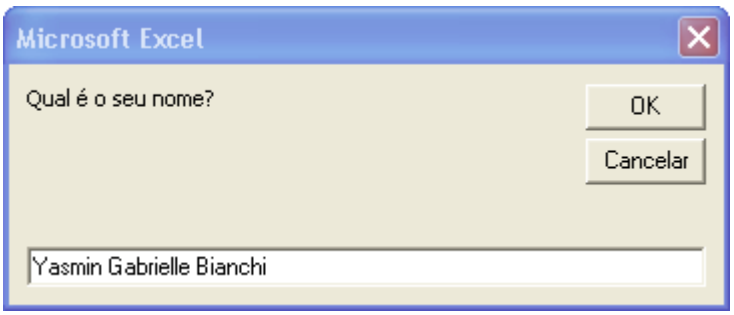
Comentários são utilizados com a finalidade de documentar o programa-fonte. Eles não são tratados pelo compilador. O símbolo utilizado para representar comentários inseridos no programa é o apóstrofo (‘).

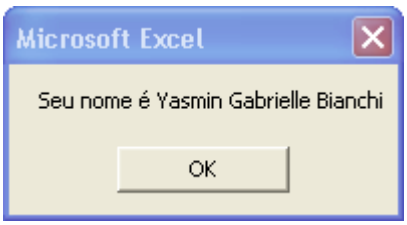
O exemplo, a seguir, apresenta a declaração de variável, o operador de atribuição, as funções de entrada e saída e comentários:

```
'O usuário introduz o seu nome e o programa o exibe numa
'caixa de mensagem.

Sub exemplo1()
    Dim nome As String        'Declaração de variável
      'Atribui o valor de retorno da função InputBox à variável nome
    nome = InputBox("Qual é o seu nome?")
    MsgBox "Seu nome é " & nome   'Exibe o conteuto da variável nome
End Sub
```

Resultado da execução:





Operadores

Aqui serão considerados os operadores aritméticos, relacionais, de concatenção, lógicos e a predência deles na avalição de uma expressão:

Operadores aritméticos

A tabela, a seguir, apresenta os símbolos e as respectivas operações e sintaxes dos operadores aritméticos:

Operador	Significado	Sintaxe
^	potenciação	r = b^e
*	multiplicação	r = n1*n2
/	divisão	r = n1/n2
\	divisão (quociente inteiro)	r = n1\n2
Mod	divisão (retorna o resto)	r = n1 Mod n2
+	soma	r = n1+n2
-	subtração	r = n1-n2

Operadores relacionais

Operadores relacionais fazem comparações, isto é, verificam a relação de magnitude e igualdade entre dois valores. São seis os operadores relacionais:

Operador	Significado
>	maior
>=	maior ou igual
<	menor
<=	menor ou igual
=	igual
<>	diferente

Operadores de concatenação

Usados para concatenar e adicionar seqüências de caracteres de duas expressões:

Operador	Significado	Sintaxe
&	concatena	r = expr1 & expr2
+	adiciona	r = expr1 + expr2

Operadores lógicos

Os operadores lógicos do VBA são:

Operador	Significado	Descrição
and	e	conjunção
or	ou	disjunção
not	não	negação
xor	exclusão	disjunção exclusiva
imp	implicação	condicional
eqv	equivalência	bicondicional

As tabelas-verdade, a seguir, expressam melhor essas operações lógicas:

&& conjunção)	| | (disjunção)	! (negação)	xor(disj.excl.)	imp(cond.)	eqv(bicond..)
0 e 0 = 0	0 ou 0 = 0	não 0 = 1	0 v 0 = 1	0 → 0 =1	0 <->0 = 1
0 e 1 = 0	0 ou 1 = 1	não 1 = 0	1 v 1 = 0	0 → 1 = 1	0 <-> 1 = 0
1 e 0 = 0	1 ou 0 = 1		1 v 0 = 1	1 → 0 = 0	1 <-> 0 = 0
1 e 1 = 1	1 ou 1 = 1		1 v 1 = 1	1 → 1 = 1	1 <-> 1 = 1

O exemplo que segue usa o operador **And** para executar uma conjunção lógica em duas expressões:

```
Dim A, B, C, R
A = 10: B = 8: C = 6:  ' Inicializa as variáveis.
R = A > B And B > C    ' Retorna True.
R = B > A And B > C    ' Retorna False.
R = A And B             ' Retorna 8 (comparação bit a bit).
```

Exemplo com operador **Or** para executar a disjunção lógica em duas expressões:

```
Dim A, B, C, R
A = 10: B = 8: C = 6:  ' Inicializa as variáveis.
R = A > B Or B > C      ' Retorna True.
R = B > A Or B > C      ' Retorna True.
R = A Or B              ' Retorna 10 (comparação bit a bit).
```

Precedência dos operadores

Uma expressão é avaliada e resolvida em uma ordem predeterminada chamada precedência de operadores.

Os operadores de várias categorias na mesma expressão são avaliados na seguinte ordem: 1º os operadores aritméticos, 2º os operadores relacionais e, por último os operadores lógicos. Os operadores relacionais são avaliados na ordem em que aparecem na expressão. Os operadores aritméticos e lógicos obedecem a seguinte ordem de precedência, da esquerda para a direita:

Operadores aritméticos:

^	- (unário)	* e /	\	Mod	+ e -

Operadores lógicos:

Not	And	Or	Xor	Eqv	Imp

Estruturas de controle

A lógica do procedimento flui através das instruções da esquerda para a direita e de cima para baixo. As instruções de controle, ou seja, as instruções que controlam a tomada de decisões e as iterações podem alterar a ordem de execução das instruções.

Estruturas de seleção ou de decisão

As instruções condicionais avaliam se uma condição é Verdadeira ou Falsa, e em seguida especificam uma ou mais instruções a serem executadas, dependendo do resultado.

⇒ **If ...Then...Else**: Desvia quando uma condição é True ou False.

⇒ **Select Case**: Seleciona um desvio de um conjunto de condições.

Seleção simples

(**If ... Then**) testa uma condição única e executa uma instrução ou um bloco de instruções.

Exemplos: a)
```
If valor < 0 then Valor = 0
```

b)
```
If valor > 5 then
      soma = valor + 30
      Valor = 0
End If
```

O exemplo em "a" de linha única não usa a instrução **End If** como ocorre com o exemplo em "b" que contém mais de uma linha de código. Portanto, a sintaxe de linha múltipla é **If ... Then ... End If**.

Seleção composta

(**If … Then … Else**) testa uma condição única e executa um entre dois blocos de instruções.

Exemplo:

```
'Verifica e informa se o usuário pode ou não obter carteira de
'habilitação.

Sub CarteiraHab()
    idade = InputBox("Insira a idade", , 15)
    If idade < 16 Then
        MsgBox "Não pode obter carteirade habilitação"
    Else
        MsgBox "Pode obter carteira de habilitação"
    End If
End Sub
```

Seleção encadeada

(**If … Then … ElseIf**) testa mais de uma condição e executa um dos vários blocos de instruções.

Exemplo:

```
'Calcula e mostra o valor do bonus com base no cargo
'e salário do funcionário.

Sub bonus()
    Dim cargo As Integer
    Dim salario As Currency, bonus As Currency
    salario = InputBox("Informe o salario: ")
    cargo = InputBox("Informe o cargo: ")
    If cargo = 1 Then
       bonus = salario * 0.15
    ElseIf cargo = 2 Then
       bonus = salario * 0.1
    ElseIf cargo = 3 Then
       bonus = salario * 0.8
    Else
       bonus = 0
    End If
MsgBox "Cargo: " & cargo & " Bonus: " & bonus
```

(No entanto, se cada instrução ElseIf testar a mesma expressão com valores diferentes é mais prático utilizar a estrutura de múltipla escolha).

Seleção de múltipla escolha

(**Select Case**) testa uma condição única e executa um dos vários blocos de instruções.

Exemplo:

```
'Calcula e mostra o valor do bonus com base no cargo
'e salário do funcionário.

Sub bonus()
    Dim cargo As Integer
    Dim salario As Currency, bonus As Currency
    salario = InputBox("Informe o salario: ")
    cargo = InputBox("Informe o cargo: ")
    Select Case cargo
        Case 1: bonus = salario * 0.15
        Case 2: bonus = salario * 0.1
        Case 3: bonus = salario * 0.08
        Case 4, 5       'Pode conter vários valores
                bonus = salario * 0.05
        Case 6 To 8     'Pode ser um intervalo de valores
                bonus = salario * 0.01
        Case Is < 12    'Pode ser comparado a outros valores
                bonus = salario * 0.005
        Case Else: bonus = 0
    End Select
    MsgBox ("Cargo: " & cargo & " Bonus: " & bonus)
End Sub
```

A estrutura **Select Case** pode ser usada em lugar da **If ... Then ... ElseIf** apenas quando a instrução **ElseIf** avaliar a mesma expressão.

Estruturas de repetição

Permitem a execução de um grupo ou bloco de instruções repetidamente. As intruções podem ser repetidas até que uma condição seja **falsa** ou até que seja **verdadeira**.
Também há loops que repetem instruções um número específico de vezes ou em cada objeto de uma coleção.

Do...Loop: Faz um loop enquanto ou até que uma condição seja *True*.
For...Next: Utiliza um contador para executar instruções um determinado número de vezes.
For Each...Next: Repete um grupo de instruções para cada objeto em uma coleção.

Teste no início

Do While ... Loop testa uma condição no início do loop e executa o **loop** enquanto a condição for **True**.

Sintaxe: **Do While** condição
Instruções
Loop

Exemplo:

```
'Lê valores para o cálculo da média aritmética,
'encerrando o processo com a entrada de valor negativo.

Sub Media()
    Dim valor As Long, soma As Long, i As Long
    valor = 0: soma = 0: i = 0
    Do While (valor >= 0)
        valor = InputBox("Insira valores para o cálculo da média." & _
                Chr(13) & "Para encerrar digite um valor negativo")
        If valor >= 0 Then
            soma = soma + valor
            i = i + 1
        End If
    Loop
    MsgBox "Média = " & (soma / i)
End Sub
```

Do Until ... Loop testa uma condição no início do loop e executa o **loop** enquanto a condição for **False**.

Sintaxe: **Do Until** condição
Instruções
Loop

Exemplo:

```
'Conta o número de vezes que o usuário executa este procedimento.

Sub conta()
    Dim soma As Integer
    soma = 0
    resp = vbYesNo
    Do Until resp = vbNo
        soma = soma + 1
        resp = MsgBox ("Deseja continuar?", vbYesNo)
    Loop
    MsgBox "Total = " & soma
End Sub
```

Teste no final

Do ... Loop While testa uma condição no final do **loop** e continua a execução enquanto a condição for **True**.

Sintaxe: **Do**
Instruções
Loop While condição

Exemplo:

```
'Converte para o sistema binário um número decimal informado pelo
usuário.

Sub Converte()
    Dim dec As Integer, bin As String
    Dim resto As Integer, sResto As String
    dec = InputBox("Informe um num.", , 19)
    Do
        resto = dec Mod 2     ' retorna o resto da divisão
        sResto = CStr(resto)  ' converte o resto para o tipo string
        bin = sResto + bin    ' concatena o resto com o conteúdo de bin
        dec = dec \ 2         ' retorna o quaciente inteiro da divisão
    Loop While dec > 0
    MsgBox "Valor em binário: " & bin
End Sub
```

Do ... Loop Until testa uma condição no final do **loop** e continua a execução enquanto a condição for **False**.

Sintaxe: **Do**
Instruções
Loop Until condição

Exemplo:

```
'Calcula e mostra o fatorial de um número fornecido pelo usuário.

Sub fatorial()
    Dim num As Integer, fat As Integer, i As Integer
    num = InputBox("Informe um num.", , 5)
    fat = 1: i = 1
    Do
        fat = fat * i
        i = i + 1
    Loop While i <= num
    MsgBox "Fatorial de " & num & ": " & fat
End Sub
```

Repetição com variável de controle

For ... Next executa as instruções do **loop** enquanto a variável contadora não atingir o valor especificado.

Exemplos:

```
'Emite sinais sonoros.

Sub nBeeps()
    Dim numBeeps As Long, i As Long
    numBeeps = InputBox("Quantos beeps?")
    For i = 1 To numBeeps
        Beep
    Next i
End Sub
```

```
'Efetua a soma dos números pares até 10.

Sub soma()
    Dim total As Integer, j As Integer
    For j = 2 To 10 Step 2
        total = total + j
    Next j
    MsgBox "O total é " & total
End Sub
```

Repetição com objetos de coleção

For Each... Next executa as instruções do **loop** para cada objeto de uma coleção.

Exemplo:

```
'Preenche com o valor 100 as células A1:D10 de Plan1
'que apresentarem valor menor do que 4.

Sub preenche()
    Dim inter As Range, c As Range
    Set inter = Worksheets(1).Range("A1:D10")
    For Each c In inter
```

```
        If c.Value < 4 Then
            c.Value = 100
        End If
    Next c
End Sub
```

Saindo de loops e procedimentos

As intruções **Exit** permitem abandonar uma estrutura de controle. Apesar dessas instruções serem convenientes, deve-se restringir seu uso uma vez que o excesso de sua utilização pode dificultar a leitura e a depuração do código.

Para sair diretamente de uma estrutura loop **For**, utiliza-se a instrução **Exit For** e para sair diretamente de um loop **Do** usa-se a instrução **Exit Do**.

As instruções **Exit Sub** e **Exit Function** podem ser usadas para abandonar um procedimento.

Matrizes

Matriz é uma coleção de variáveis que apresenta uma estrutura de dados multidimensional. Cada elemento da matriz pode ser distinguido de outros elementos por um ou mais índices inteiros.

Matriz unidimensional

A matriz unidimensional ou **vetor** é exatamente uma seqüência linear de elementos armazenados consecutivamente na memória.

Declaração de matrizes

Exemplos:

```
1.  Dim matNum (20) As Integer
2.  Dim soma (1 To 15) As Integer
3.  Dim valor (100 To 120) As String
```

No primeiro exemplo acima o índice da matriz vai de 0 a 20 – contém 21 elementos, (sem a especificação da opção **Option Base 1** na seção de declaração do módulo).

No segundo e terceiro exemplos os índices variam de 1 a 15 e de 100 a 120, respectivamente.
O tamanho da matriz é determinado pelo número das suas dimensões e pelos limites superiores e inferiores dos índices de cada dimensão.

Atribuição

O exemplo, a seguir, atribui valores a três elementos da matriz diaSemana:

```
diaSemana(1) = "Segunda-feira"
diaSemana(2) = "Terça-feira"
diaSemana(3) = "Quarta-feira"
```

A instrução seguinte recupera e exibe o conteúdo do sexto elemento dessa matriz:

```
MsgBox diaSemana(5)
```

O exemplo a seguir atribui o vocábulo inicial "Bom" para todos os elementos da matriz.
O comando **Debug.Print** envia a saída para a janela “verificação imediata” que pode ser aberta a partir do menu Exibir do programa editor Visual Basic.

```
Sub semana()
    Dim diaSemana(6) As String
    Dim i As Integer
    For i = 0 To 6
        diaSemana(i) = "Bom"
        Debug.Print diaSemana(i)
    Next i
End Sub
```

Matriz multidimensional

No Visual Basic, pode-se declarar matrizes com até 60 dimensões. Por exemplo, a instrução a seguir declara uma matriz bidimensional de 5 por 10.

```
Dim sngMulti (1 To 5, 1 To 10) As Single
```

Utilize instruções aninhadas **For...Next** para processar as matrizes multidimensionais.
O procedimento, a seguir, preenche cada elemento da matriz bidimensional com o valor baseado na sua localização dentro da matriz:

```
Sub MatrizMulti()
    Dim i As Integer, j As Integer
    Dim matriz(1 To 5, 1 To 10) As Single
    ' Preenche a matriz com valores e os imprime.
    For i = 1 To 5
        For j = 1 To 10
            matriz(i, j) = i * j
            Debug.Print matriz(i, j)
        Next j
    Next i
End Sub
```

Matriz fixa

Na linha de código, a seguir, uma matriz de tamanho fixo é declarada como uma matriz **Integer** com 11 linhas e 11 colunas:

```
Dim ExMatriz (10, 10) As Integer
```

O primeiro argumento representa as linhas; o segundo argumento representa as colunas.

Matriz dinâmica

Declara-se uma matriz dinâmica deixando vazios os parênteses, como mostra o exemplo a seguir:

```
Dim num( ) As integer
```

Posteriormente, no procedimento, pode ser especificado o número de elementos através da instrução **ReDim**.

Segue um exemplo de declaração e redimensionamento de uma matriz dinâmica:

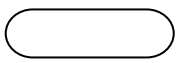
```
Sub redimatriz()
      Dim num() As Integer    ' declara uma matriz dinâmica
      ReDim num(5)            ' faz a alocação de cinco elementos
      For i = 1 To 5          ' faz o loop 5 vezes
         num(i) = i           ' inicializa a matriz
      Next i
'A próxima instrução redimensiona a matriz e apaga os elementos.
      ReDim num(10)           ' redimensiona para 10
      For i = 1 To 10         ' faz o loop 10 vezes
         num(i) = i           ' inicializa a matriz
      Next i
'A instrução a seguir redimensiona a matriz sem apagar os elementos.
       ReDim Preserve num(15)  ' redimensiona para 15
End Sub
```

Diagrama de blocos

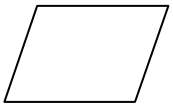
O primeiro passo ao elaborar-se um programa é definir um plano claro e completo daquilo que o aplicativo deverá realizar. A melhor maneira de faze-lo é preparar um diagrama de blocos ou fluxograma representativo da seqüência de etapas a serem executadas pelo computador.

O diagrama de blocos, também chamado de fluxograma, é a representação gráfica de um algoritmo e objetiva mostrar a forma ou a seqüência de raciocínio e as operações envolvidas para a resolução de um problema.

Os principais símbolos gráficos utilizados em um fluxograma são os seguintes:

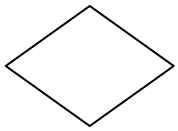


Teminal
O ponto de início, término ou interrupção de um programa.



Entrada/Saída
Leitura ou gravação de dados.

Processamento
Um grupo de instruções que executa uma função de processamento do programa.



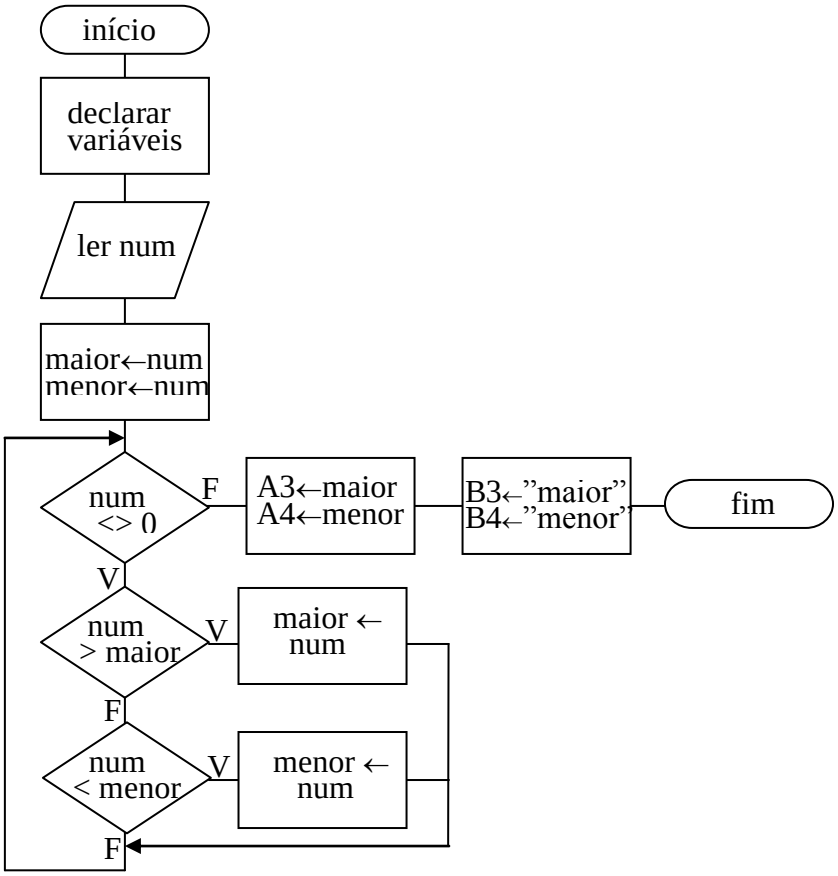
Decisão
Indica a possibilidade de desvios para outros pontos do programa dependendo do resultado de operações de comparação.

Conector
Uma entrada ou saída de ou para outra parte do fluxograma.

Especificação do problema

Achar o maior e menor número de uma série de números positivos. O último número da série contém valor 0 (zero) que indica fim de processo.

Fluxograma v.1



CódigoVBA v.1

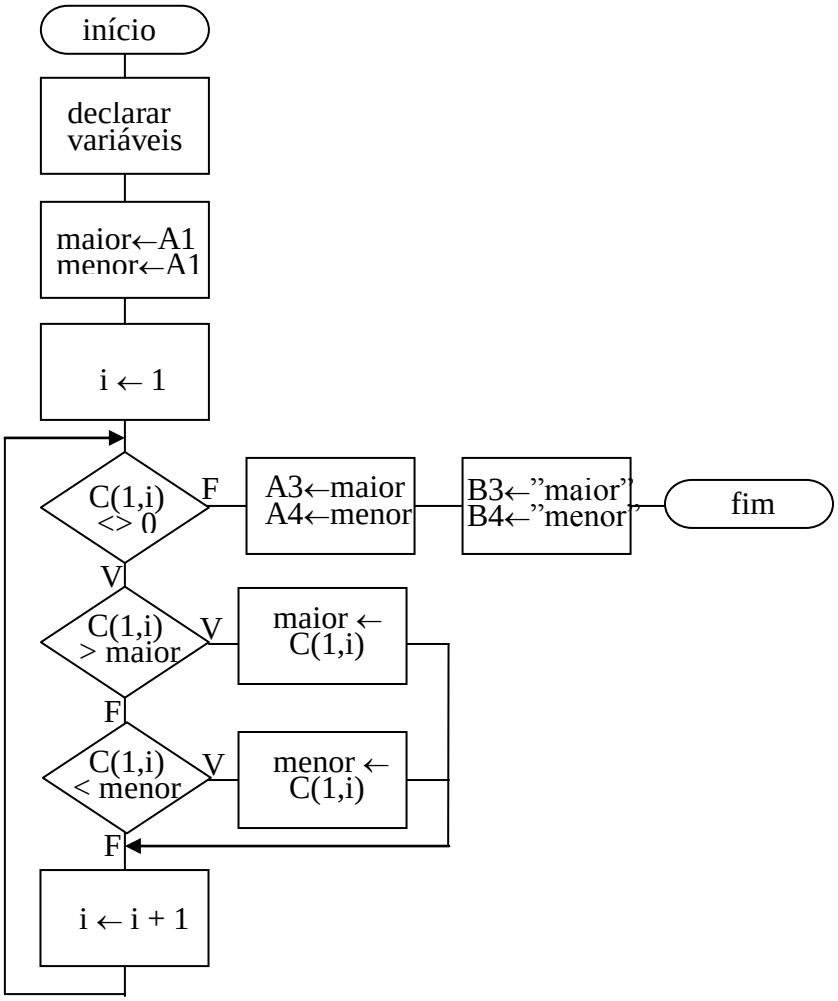
```
Sub MaiorMenor()
    Dim maior As Integer, menor As Integer
    Dim num As Integer
    num = InputBox("Entre com um num.", "Num.inteiro")
    maior = num: menor = num
    Do While (num <> 0)
        If num > maior Then
            maior = num
        Else
            If num < menor Then
                menor = num
            End If
        End If
        num = InputBox("Entre com um num.", "Num.inteiro")
    Loop
    MsgBox "Maior: " & maior & " Menor: " & menor
End Sub
```

A seguir é apresentado uma nova versão do procedimento, considerando os valores de entrada armazenados na primeira linha a partir da célula A1 da planilha “Plan1”, como mostrado na figura que segue. O resultado será exibido nas céluas A3 e A4 acompanhado das respectivas descrições colocadas nas células B3 e B4.

Valores armazenados em Plan1:

	A	B	C	D	E	F	G	H
1	23	45	31	12	10	42	18	0

Fluxograma v.2



CódigoVBA v.2

```
Sub MaiorMenor()
        Worksheets("plan1").Activate
        Dim maior As Integer, menor As Integer
        Dim num As Integer, i As Integer
        i = 1
        maior = Range("A1"): menor = Range("A1")
        Do While (Cells(1, i) <> 0)
              If (Cells(1, i) > maior) Then
                    maior = Cells(1, i)
              Else
                    If (Cells(1, i) < menor) Then
                              menor = Cells(1, i)
                    End If
               End If
              i = i + 1
        Loop
        Range("A3") = maior
        Range("A4") = menor
        Range("B3") = "Maior"
        Range("B4") = "Menor"
End Sub
```

Saída do processamento:

	A	B	C	D	E	F	G	H
1	23	45	31	12	10	42	18	0
2								
3	45	Maior						
4	10	Menor						

Bibliografia

- Programando em Microsoft Excel 7-Visual Basic. Sao Paulo : Makron Books, 1996. xxvii, 359p.
- CAPRON, H. L. Introdução à informática. São Paulo : Pearson Education, 2004.
- FORBELLONE, André Luiz Villar; EBERSPACHER, Henri Frederico. Lógica de programação : a construção de algoritmos e estruturas de dados. 2.ed. São Paulo : Makron Books, 2000. 197p.
- UCCI, Waldir; SOUSA, Reginaldo Luiz; KOTANI, Alice Mayumi, et al. . Lógica de programação : os primeiros passos. 8.ed. Sao Paulo : Erica, 1999. 339p.

Eletrônica

- Excel/VBA: http://www.excel-vba-access.com/