Instrução Do...Loop (Visual Basic)

Artigo • 06/04/2023

Repete um bloco de instruções enquanto uma condição `Boolean` é `True` ou até que a condição se torne `True`.

Sintaxe

VB

```
Do [ { While | Until } condition ]
    [ statements ]
    [ Continue Do ]
    [ statements ]
    [ Exit Do ]
    [ statements ]
Loop
' -or-
Do
    [ statements ]
    [ Continue Do ]
    [ statements ]
    [ Exit Do ]
    [ statements ]
Loop [ { While | Until } condition ]
```

Partes

Termo	Definição
`Do`	Obrigatórios. Inicia a definição do loop `Do`.
`While`	Não poderá ser dado se `Until` for usado. Repita o loop `condition` até que seja `False`.
`Until`	Não poderá ser dado se `While` for usado. Repita o loop `condition` até que seja `True`.
`condition`	Opcional. Expressão `Boolean`. Se `condition` for `Nothing`, o Visual Basic o tratará como `False`.
`statements`	Opcional. Uma ou mais instruções que são repetidas enquanto, ou até que `condition` seja `True`.

Termo	Definição
`Continue Do`	Opcional. Transfere o controle para a próxima iteração do loop `Do`.
`Exit Do`	Opcional. Transfere o controle para fora do loop `Do`.
`Loop`	Obrigatórios. Finaliza a definição do loop `Do`.

Comentários

Use uma estrutura `Do...Loop` quando quiser repetir um conjunto de instruções um número indefinido de vezes até que uma condição seja atendida. Se você quiser repetir as instruções um número definido de vezes, a instrução For...Next geralmente é uma opção melhor.

Você pode usar `While` ou `Until` para especificar `condition`, mas não ambos. Se você não fornecer nenhum deles, o loop continuará até que um Exit transfira o controle para fora do loop.

Você pode testar `condition` apenas uma vez, no início ou no final do loop. Se você testar `condition` no início do loop (na instrução `Do`), o loop poderá não ser executado nem uma vez. Se você testar no final do loop (na instrução `Loop`), o loop sempre será executado pelo menos uma vez.

A condição geralmente resulta de uma comparação de dois valores, mas pode ser qualquer expressão avaliada como um valor de Tipo de dados booliano (`True` ou `False`). Isso inclui valores de outros tipos de dados, como tipos numéricos, que foram convertidos em `Boolean`.

Você pode aninhar loops `Do` colocando um loop dentro de outro. Você também pode aninhar diferentes tipos de estruturas de controle entre si. Para obter mais informações, confira Estruturas de controle aninhadas.

> **Observação**
>
> A estrutura `Do...Loop` oferece mais flexibilidade do que a **instrução While...End While** porque permite que você decida se deseja encerrar o loop quando `condition` parar de ser `True` ou quando ele se tornar `True` pela primeira vez. Ele também permite que você teste `condition` no início ou no final do loop.

Exit Do

A instrução Exit Do pode fornecer uma forma alternativa de sair de um `Do…Loop`. `Exit Do` transfere o controle imediatamente para a instrução que segue a instrução `Loop`.

`Exit Do` geralmente é usado após avaliação de alguma condição, por exemplo, em uma estrutura `If...Then...Else`. Talvez você queira sair de um loop se detectar uma condição que torna desnecessário ou impossível continuar iterando, como um valor errôneo ou uma solicitação de encerramento. Um uso de `Exit Do` serve para testar uma condição que pode causar um *loop infinito*, que é um loop que pode ser executada várias vezes ou até mesmo infinitamente. Você pode usar `Exit Do` para ignorar o loop.

Você pode incluir qualquer número de instruções `Exit Do` em qualquer lugar em um `Do…Loop`.

Quando usado em loops `Do` aninhados, `Exit Do` transfere o controle para fora do loop mais interno e para dentro do próximo nível mais alto de aninhamento.

Exemplo 1

No exemplo a seguir, as instruções no loop continuam a ser executadas até que a variável `index` seja maior que 10. A cláusula `Until` está no final do loop.

VB

```
Dim index As Integer = 0
Do
    Debug.Write(index.ToString & " ")
    index += 1
Loop Until index > 10

Debug.WriteLine("")
' Output: 0 1 2 3 4 5 6 7 8 9 10
```

Exemplo 2

O exemplo a seguir usa uma cláusula `While` em vez de uma cláusula `Until`, e `condition` é testado no início do loop em vez de no final.

VB

```
Dim index As Integer = 0
Do While index <= 10
    Debug.Write(index.ToString & " ")
    index += 1
Loop

Debug.WriteLine("")
' Output: 0 1 2 3 4 5 6 7 8 9 10
```

Exemplo 3

No exemplo a seguir, `condition` interrompe o loop quando a variável `index` é maior que 100. No entanto, a instrução `If` no loop faz com que a instrução `Exit Do` interrompa o loop quando a variável de índice for maior que 10.

VB

```
Dim index As Integer = 0
Do While index <= 100
    If index > 10 Then
        Exit Do
    End If

    Debug.Write(index.ToString & " ")
    index += 1
Loop

Debug.WriteLine("")
' Output: 0 1 2 3 4 5 6 7 8 9 10
```

Exemplo 4

O exemplo a seguir lê todas as linhas em um arquivo de texto. O método OpenText abre o arquivo e retorna um StreamReader que lê os caracteres. Na condição `Do...Loop`, o método Peek do `StreamReader` determina se há caracteres adicionais.

VB

```
Private Sub ShowText(ByVal textFilePath As String)
    If System.IO.File.Exists(textFilePath) = False Then
        Debug.WriteLine("File Not Found: " & textFilePath)
    Else
        Dim sr As System.IO.StreamReader =
```

```
System.IO.File.OpenText(textFilePath)

        Do While sr.Peek() >= 0
            Debug.WriteLine(sr.ReadLine())
        Loop

        sr.Close()
    End If
End Sub
```

Confira também

- Estruturas de Loop
- Instrução For...Next
- Tipo de dados booliano
- Estruturas de Controle Aninhadas
- Instrução Exit
- Instrução While...End While