

# FUNÇÕES INTERNAS

---

## ABS

---

**Categoria:** Função Matemática

**Finalidade:** Retorna o valor absoluto de um número.

**Sintaxe:** ABS(Número)

**Em que:**

- Número - Qualquer valor numérico válido. Se Número contém Null, o valor retornado pela função será Null também. Caso Número seja igual a zero, este será o valor de retorno da função.

**Exemplo:**

```
Teste = ABS(2.99)           `Teste = 2.99
Teste = ABS(-2.99)        `Teste = 2.99
```

---

## Array

---

**Categoria:** Funções de Array

**Finalidade:** Retorna um dado Variant contendo um Array.

**Sintaxe:** Array(Lista de Argumentos)

**Em que:**

- Lista de Argumentos - Consiste em uma série de argumentos separados por vírgulas. Se nenhum argumento for fornecido à função, será criado um array de comprimento zero.

**Exemplo:**

```
MeuArray("Jan","Fev","Mar","Abr")      `MeuArray(2) = "Fev"
MeuArray(1,3,5,7)                      `MeuArray(3) = 5
```

---

## Asc

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Retorna o código ASCII corresponde quente do primeiro caractere da string indicada.

**Sintaxe:** ASC(String)

**Em que:**

- String - O argumento string é qualquer expressão de caracteres válida. Se a string especificada não contiver caracteres, ocorrerá um erro de Run-Time.

**Exemplo:**

```
Teste = Asc("A")           `Teste = 65
Teste = Asc("a")          `Teste = 97
```

```
Teste = Asc("ASCII")           `Teste = 65
```

---

## Atn

---

**Categoria:** Funções Matemáticas

**Finalidade:** Calcula o Arco Tangente de um número.

**Sintaxe:** Atn(Número)

**Em que:**

- Número - A função Atn retorna o número de radianos referentes ao arco tangente do ângulo especificado.

**Exemplo:**

```
Teste = Atn(30)                 `Teste = 1.53747533091665  
Teste = Atn(45)                 `Teste = 1.54857776146818
```

---

## Cbool

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão em Boolean.

**Sintaxe:** CBool(Expressão)

**Em que:**

- Expressão - Se o valor da expressão resultante for zero, o valor False será retornado pela função Cbool; caso contrário, a resposta será True.

**Exemplo:**

```
Primeiro = 10  
Segundo = 10  
Valor = 0
```

```
Teste = Cbool(Primeiro = Segundo)   `Teste = True  
Teste2 = Cbool(Valor)                `Teste2 = False
```

---

## Cbyte

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão em Byte.

**Sintaxe:** CByte(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido.

**Exemplo:**

```
Dim Teste as Double
```

```
Teste = 222.3322  
MeuByte = CByte(Teste)           `MeuByte = 222
```

---

## CCur

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão em Currency.

**Sintaxe:** CCur(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido ou expressão string.

**Exemplo:**

Dim Dado as Double

Dado = 656.123456

Dado = Dado \* 2

MinhaMoeda = CCur(Dado)

`MinhaMoeda = 1312.2469

---

## CDate

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão no tipo de dados Date.

**Sintaxe:** CDate(Data)

**Exemplo:**

Hoje = "June 18, 1999"

DataCurta = CDate(Hoje)

`DataCurta = 18/06/99

---

## CDBl

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão em Double.

**Sintaxe:** CDBl(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido ou expressão string.

**Exemplo:**

Dim MinhaMoeda As Currency

MinhaMoeda = 255.123456

Dado = CDBl(MinhaMoeda)

`Dado = 255.1235

---

## CDec

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão em Decimal.

**Sintaxe:** CDec(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido ou expressão string.

**Exemplo:**

```
Dim ValorTeste
MeuValor = 3.05E+15
ValorTeste = CDec(MeuValor)      'ValorTeste= 3050000000000000
```

**Choose**

**Categoria:** Funções de Verificação

**Finalidade:** Selecciona e retorna um valor de uma lista de argumentos existente em função do índice passado.

**Sintaxe:** Choose(index, choice-1[, choice-2, ... [,choice-n]])

**Em que:**

- **Index** - Expressão numérica ou campo que resulta em um valor entre 1 e o número de escolhas disponíveis.
- **Choice** - Expressão Variant contendo uma das possíveis escolhas.

**Exemplo:**

```
Teste = Escolha(2)      'Teste = "Valor2"
```

**Função Escolha:**

```
Function Escolha(Indice As Integer)
    Escolha = Choose(Indice, "Valor1", "Valor2", "Valor3")
End Function
```

**Chr**

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Retorna o caractere correspondente ao código ASCII informado.

**Sintaxe:** Chr(Código ASCII)

**Exemplo:**

```
Teste = Chr(65)      'Teste = A
Teste = Chr(97)      'Teste = a
Teste = Chr(64)      'Teste = @
```

**CInt**

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão no tipo de dados Inteiro.

**Sintaxe:** CInt(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido ou expressão string.

### Exemplo:

```
Dim Duplo As Double
Duplo = 8765.4321
Teste = CInt(Duplo)           `Teste = 8765
```

---

## CLng

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão no tipo de dados Longo.

**Sintaxe:** CLng(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido ou expressão string.

### Exemplo:

```
Dim Dado As Double
Dado = 65432.123
Teste = CLng(Dado)           `Teste = 65432
```

---

## Command

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Retorna o argumento informado na linha de comando utilizado na inicialização do Visual Basic ou de um programa executável desenvolvido nesta linguagem.

**Sintaxe:** Command

### Exemplo:

Inicie a criação de um aplicativo qualquer e inclua os comandos abaixo na procedure Form\_Load. A seguir, selecione o comando **Options...** no menu **Tools**, escolhendo em seguida a aba **Advanced**. Inclua o seguinte argumento na caixa de texto Command Line Arguments: *Start*. A seguir, execute seu aplicativo com e sem este argumento e veja o que acontece.

```
Dim Msg

If Command = "" Then
    Msg = "Não existe uma linha de comando"
Else
    Msg = "A linha de comando é: " & Command
End If

MsgBox Msg
```

---

## Cos

---

**Categoria:** Funções Matemáticas

**Finalidade:** Calcula o cosseno de um ângulo.

**Sintaxe:** Cos(Número)

**Em que:**

- Número - Pode ser qualquer expressão numérica válida que expresse um ângulo em radianos.

**Exemplo:**

```
Const PI = 3.14159265
```

```
Teste = Cos(PI)
```

```
\Teste = -1
```

```
Teste = Cos(PI/2)
```

```
\Teste = 1.79489651491878E-09
```

---

## CSng

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão no tipo de dados Single.

**Sintaxe:** CSng(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido ou expressão string.

**Exemplo:**

```
Dim Teste As Double
```

```
Teste = 22.12345678
```

```
Teste = CSng(Teste)
```

```
\Teste = 22.12346
```

---

## CStr

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão no tipo de dados String.

**Sintaxe:** CStr(Expressão)

**Em que:**

- Expressão - O argumento da função poderá ser qualquer valor numérico válido ou expressão string.

**Exemplo:**

```
Dim Dado As Double
```

```
Dado = 555.4321
```

```
Teste = CStr(Dado)
```

```
\Teste = "555.4321"
```

---

## CurDir

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Retorna a pasta (ou diretório) atual.

**Sintaxe:** CurDir[(Drive)]

**Em que:**

- Drive - Expressão string que especifica uma unidade de disco existente. Se a unidade não for especificada ou se a string for de tamanho zero, a função CurDir retornará o caminho (Path) para o drive atual.

**Exemplo:**

```
Teste = CurDir           `Teste = C:\WINDOWS
```

**CVar**

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Converte uma expressão no tipo de dados Variant.

**Sintaxe:** CVar(Expressão)

**Em que:**

Expressão - Qualquer valor numérico válido ou expressão string.

**Exemplo:**

```
Dim Inteiro As Integer
Inteiro = 2222
Teste = CVar(Inteiro & "000")           `Teste = "2222000"
```

**CVErr**

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Retorna o número do erro especificado pelo usuário

**Sintaxe:** CVErr(Número do Erro)

Você pode utilizar a função CVErr para definir seus próprios números de erro. Como você já deve ter notado ao longo de nosso estudo, sempre que ocorre um erro em tempo de execução, o VB exibe um bloco de diálogo que mostra o número do erro ocorrido e a mensagem correspondente a esse erro. Você também pode criar suas próprias mensagens de erro e com isso controlar o comportamento de seu aplicativo no caso da ocorrência de situações indesejadas. Você pode utilizar a função **IsError** para avaliar o erro ocorrido e tomar as devidas providências.

**Exemplo:**

```
Dado = TxtIn.TEXT           `Entrada de Dado
Teste = CalculoGeral(Dado)

If IsError(Teste) Then     `Se Teste é um Número de erro
    Teste = CStr(Teste)    `Converto no tipo de dados String
End If

TxtOut.TEXT = Teste
```



ww	Week (Semana)
h	Hour (Hora)
n	Minute (Minuto)
s	Second (Segundo)

### Exemplo:

```
REM Acresce três anos à data atual
Teste = DateAdd("yyyy", 3, #10/04/99#) `Teste = 04/10/02
```

```
REM Diminui 6 meses da data atual
Teste = DateAdd("m", -6, #10/04/99#) `Teste = 04/04/99
```

```
REM Soma uma semana à data atual
Teste = DateAdd("ww", 1, #10/4/99#) `Teste = 11/10/99
```

Para efeito de cálculo, o VB considera os anos bissextos, de modo que o mês de fevereiro será considerado com 28 ou 29 dias de acordo com a data especificada e o intervalo de tempo informado.

---

## DateDiff

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o intervalo de tempo decorrido entre duas datas de acordo com a unidade de tempo especificada.

**Sintaxe:** DateDiff(Intervalo, Data1, Data2[, firstdayofweek[, firstweekofyear]])

**Em que:**

- Intervalo - Expressão do tipo string indicando o intervalo de tempo que será calculado entre Data1 e Data2. Pode ser horas, dias, semanas, meses, anos, etc., conforme especificado na seção ajustes, logo em seguida.
- Data1, Data2 - As duas datas entre as quais será calculado o intervalo de tempo decorrido.
- Firstdayofweek - Uma constante que define o primeiro dia da semana. Caso você não especifique, o domingo será considerado como primeiro dia. Veja a seção ajustes, logo em seguida, para maiores detalhes.
- Firstweekofyear - Esta constante define a primeira semana do ano. Se você não especificar este argumento, será considerada como a primeira do ano a semana em que ocorre o dia 1º de janeiro.

**Ajustes:**

Para Intervalo:

Ajuste	Descrição
yyyy	Year (Ano)

q	Quarter (Trimestre)
m	Month (Mês)
y	Day of year (Dia do Ano)
d	Day (Dia)
w	Weekday (Dia da Semana)
ww	Week (Semana)
h	Hour (Hora)
n	Minute (Minuto)
s	Second (Segundo)

### Para Firstdayofweek:

Ajuste	Descrição
0	O atual do sistema
1	Domingo (default)
2	Segunda
3	Terça
4	Quarta
5	Quinta
6	Sexta
7	Sábado

### Para Firstweekofyear:

Ajuste	Descrição
0	O atual do sistema (se houver)
1	A semana em que ocorre dia 1º de janeiro
2	A primeira semana que tem pelo menos 4 dias do novo ano
3	A primeira semana que contiver sete dias do novo ano

### Exemplo:

REM Dias decorridos de 1/1/99 até 4/10/99:

```
Teste = DateDiff("d", #1/1/99#, #10/4/99#) `Teste = 276
```

REM Horas decorridas entre 1/10/99 até 4/10/99:

```
Teste = DateDiff("h", #10/1/99#, #10/4/99#) `Teste = 72
```

Se Data2 for menor que a Data1, o número de intervalos decorridos será negativo.

A opção Firstdayofweek afeta os cálculos efetuados com base nos intervalos de tempo "w" (dia da semana) e "ww" (Semana).

---

### DatePart

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna a parte especificada de uma determinada data.

**Sintaxe:** DatePart(Intervalo, Data[, Firstdayofweek[, Firstweekofyear]])

**Em que:**

- Intervalo - Expressão do tipo string indicando o intervalo de tempo que será retornado pela função. Veja a seção Ajustes em seguida.
- Data - A data a ser avaliada pela função.
- Firstdayofweek - Uma constante que define o primeiro dia da semana. Caso você não especifique, o domingo será considerado como primeiro dia. Veja a seção ajustes, logo em seguida, para maiores detalhes.
- Firstweekofyear - Esta constante define a primeira semana do ano. Se você não especificar este argumento, será considerada como a primeira do ano a semana em que ocorre o dia 1º de janeiro.

**Ajustes:**

Esta função utiliza os mesmos ajustes da função **DateDiff**.

**Exemplo:**

```
REM Calcula a semana atual da data indicada
Teste = DatePart("ww", #10/4/99#)           `Teste = 41
```

---

### DateSerial

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna a data para o dia, mês e ano especificados.

**Sintaxe:** DateSerial(Ano, Mês, Dia)

**Em que:**

- Ano - Número entre 100 e 9999 inclusive.
- Mês - Qualquer expressão numérica.
- Dia - Qualquer expressão numérica.

**Exemplo:**

```
REM Retorna a data de 1º de maio de 1999
Teste = DateSerial(99, 5, 1)               `Teste = 01/05/99
```

---

### DateValue

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna uma data.

**Sintaxe:** DateValue(Data)

**Em que:**

- Data - Expressão string representando uma data. O VB pode manipular datas que vão de 1/1/100 até 31/12/9999! Você também pode utilizar qualquer expressão que represente data, hora ou ambas.

**Exemplo:**

```
REM Converte uma string numa data
Teste = DateValue("6/22/99")           `Teste = 22/06/99
```

---

**Day**

---

**Categoria:** Funções de Data e Hora**Finalidade:** Retorna um número de 1 a 31 representando o dia do mês.**Sintaxe:** Day(Data)**Exemplo:**

```
Teste = Day(#4/10/99#)                 `Teste = 10
```

---

**DDB**

---

**Categoria:** Funções Financeiras**Finalidade:** Calcula a depreciação de um ativo pelo método do Declínio de Balanço.**Sintaxe:** DDB(Cost, Salvage, Life, Period[, Factor])**Em que:**

- Cost - Custo inicial do bem.
- Salvage - Valor do bem ao final de sua vida útil.
- Life - Vida útil do bem.
- Period - Período de depreciação.
- Factor - Taxa de declínio do balanço.

**Exemplo:**

```
Teste = DDB(1000, 100, 5, 1)           `Teste = 400
```

---

**Dir**

---

**Categoria:** Funções de Sistema Operacional**Finalidade:** Retorna o nome dos arquivos ou diretórios que combinam com o padrão especificado ou atributo de arquivo, ou volume de um drive.**Sintaxe:** Dir[(Pathname[, Attributes])]**Em que:**

- Pathname - Expressão string que especifica o nome do arquivo. Essa string também pode conter o caminho completo (drive e diretório) do arquivo procurado. Se o caminho não for encontrado, Null será retornado.
- Attributes - Constante ou expressão numérica que indica os atributos do arquivo selecionado. Se omitido, somente serão retornados os arquivos normais, ou seja, arquivos ocultos de sistema e diretórios não serão exibidos.

### Ajustes:

Ajuste	Descrição
0	Normal
2	Hidden (Oculto)
4	System (Sistema)
8	Volume
16	Diretório

### Exemplo:

```

Teste = Dir("C:\WINDOWS\WIN.INI")      'Teste = "WIN.INI"
Para múltiplos arquivos:
Teste = Dir("C:\WINDOWS\*.INI")
List1.AddItem Teste

Do While Teste <> Empty
    Teste = Dir
    List1.AddItem Teste
Loop

```

Quando você utiliza caracteres curinga como \* e ?, o VB retorna o primeiro nome de arquivo que atende às condições de busca indicadas na função Dir. A partir daí, basta chamar a função Dir sem argumentos para que os próximos arquivos que atendem à condição proposta sejam exibidos.

Quando não existirem mais arquivos, a função Dir retornará uma string de comprimento zero.

## DoEvents

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Interrompe momentaneamente a execução de seu aplicativo para que o sistema operacional possa processar outros eventos pendentes.

**Sintaxe:** DoEvents( )

### Exemplo:

```

For a = 1 To 30000
    If a Mod 1000 = 0 Then      'Se o loop foi repetido 1000 vezes.
        DoEvents              'Passa o controle para o sistema operacional.
    End If
Next a

```

Quando seu aplicativo realiza tarefas muito demoradas, o sistema operacional deixa de processar a ocorrência de alguns eventos, que ficam pendentes em uma fila de execução.

É mais ou menos o que ocorre quando você dá dois cliques sobre o ícone do VB para inicializá-lo. Enquanto o seu micro vai carregando este aplicativo, as teclas que você está acionando e os cliques de mouse dados sobre a tela são ignorados pelo Windows. Depois que a sua aplicação já está carregada é que o Windows processa os eventos pendentes.

A função DoEvents também retorna o número de formulários abertos no Windows.

---

## Environ

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Retorna a string associada a uma variável de ambiente do sistema operacional (como Path e Prompt, por exemplo).

**Sintaxe:** Environ({Envstring | Número})

**Em que:**

- Envstring - Expressão string contendo o nome da variável de ambiente.
- Número - Expressão numérica correspondente à ordem da variável na tabela de variáveis de ambiente.

**Exemplo:**

```
Teste = Environ("PATH")
REM Teste = C:\WINDOWS;C:\WINDOWS\COMMAND;C:\;C" :\DOS;C:\UTIL

Teste = Environ("TEMP")      `Teste = C:\WINDOWS\TEMP
```

Se a variável de ambiente especificada não for encontrada, a função Environ retornará uma string de comprimento zero.

---

## Error

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Retorna a mensagem de erro que corresponde a um dado número de erro.

**Sintaxe:** Error[(Errornumber)]

**Em que:**

- Errornumber - Qualquer número de erro válido para o Visual Basic. Um número de erro corresponde ao valor da propriedade Number do objeto Err, e seus valores válidos vão de 0 a 65535, inclusive. Quando combinado com a propriedade Name do objeto Err, esse número representa uma mensagem de

erro particular. Se Errornumber é um número de erro válido, porém indefinido, a função Error retorna a string definida pelo usuário. Se Errornumber não é um número de erro válido, um erro ocorre. Se Errornumber é omitido, a mensagem correspondente ao erro de Run-time mais recente é exibida. Se nenhum erro de Run-time ocorreu recentemente ou o número do erro for 0, a função Error retornará uma string de comprimento zero.

**Exemplo:**

```
Teste = Error(5)           `Teste = Invalid procedure call
Teste = Error(6)         `Teste = Overflow
Teste = Error(7)         `Teste = Out of memory
```

**Para erros indefinidos:**

```
Teste = Error(4)
Rem Teste = Application-defined or object-defined error
```

---

**Exp**

---

**Categoria:** Funções Matemáticas

**Finalidade:** Retorna o valor de  $e$  (base de um logaritmo natural) elevado a uma potência qualquer.

**Sintaxe:** Exp(Número)

**Em que:**

- Número - Qualquer expressão numérica válida.

**Exemplo:**

```
Teste = Exp(1)           `Teste = 2.71828182845905
Teste = Exp(2)           `Teste = 7.38905609893065
Teste = Exp(-2)          `Teste = 0.13533528323661
```

O valor de  $e$  é aproximadamente 2,718282. Se o valor do número for superior a 709,782712893, ocorrerá um erro de overflow.

---

**FileDateTime**

---

**Categoria:** Funções de Acesso a Arquivos

**Finalidade:** Retorna a data e hora em que o arquivo foi criado ou modificado pela última vez.

**Sintaxe:** FileDateTime(Pathname)

**Em que:**

- Pathname - Expressão string que especifica o nome do arquivo. Essa string também pode conter o caminho completo (drive e diretório) do arquivo procurado.

**Exemplo:**

```
Teste = FileDateTime("C:\WINDOWS\WIN.COM")
REM Teste = 15/05/98 20:01:00
```

---

## FileLen

---

**Categoria:** Funções de Acesso a Arquivos

**Finalidade:** Retorna um valor Long contendo o tamanho do arquivo especificado em bytes.

**Sintaxe:** FileLen(Pathname)

**Em que:**

- Pathname - Expressão string que especifica o nome do arquivo. Essa string também pode conter o caminho completo (drive e diretório) do arquivo procurado.

**Exemplo:**

```
Teste = FileLen("C:\WINDOWS\WIN.COM") `Teste = 25271
```

---

## Fix

---

**Categoria:** Funções Matemáticas

**Finalidade:** Retorna a parte inteira de um número.

**Sintaxe:** Fix(Número)

**Em que:**

- Número - Qualquer expressão numérica válida. Se número contém Null, o valor de retorno será Null também.

**Exemplo:**

```
Teste = Fix(12.556)           `Teste = 12
Teste = Fix(-12.556)        `Teste = -12
Teste = Fix(1.6)            `Teste = 1
```

---

## Format

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Formata uma expressão de acordo com as instruções definidas pelo usuário.

**Sintaxe:** Format(Expressão[, Formato[, Firstdayofweek[, Firstweekofyear]]])

**Em que:**

- Expressão - Qualquer expressão válida.
- Formato - Uma expressão de formato definida pelo usuário. Veja na seção Ajustes em seguida, os formatos existentes.
- Firstdayofweek - Uma constante que define o primeiro dia da semana. Veja a função DateDiff para maiores detalhes.
- Firstweekofyear - Esta constante define a primeira semana do ano. Veja a função DateDiff para maiores detalhes.

**Ajustes:**

## Números:

Nome do Formato	Descrição
General Number	Exibe o número como ele é, ou seja, sem separadores de milhares.
Currency	Exibe o número com separadores de milhares, símbolo de moeda e dois dígitos à direita do ponto decimal.
Fixed	Exibe pelo menos um dígito à esquerda e dois à direita do ponto decimal.
Standard	Exibe o número com separadores de milhares, com pelo menos um dígito à esquerda e dois à direita do ponto decimal.
Percent	Exibe o número multiplicado por 100 e com o símbolo de porcentagem(%) exibido à sua direita. Sempre exibe duas casas decimais.
Scientific	Usa a notação científica padrão.
Yes/No	Exibe "No" se o número for zero. Caso contrário, exibe "Yes".
True/False	Exibe "False" se o número for zero. Caso contrário, exibe "True".
On/Off	Exibe "Off" se o número for zero. Caso contrário, exibe "On".

## Data e Hora:

Nome do Formato	Descrição
General Date	Para números reais exibe a data e hora, como em 22/6/96 08:33 PM. Se o número não possuir nenhuma parte fracionária, exibe somente a data, como em 22/6/96. Se não houver parte inteira, exibe apenas a hora, como em 08:33 PM. A exibição da data obedece aos ajustes atuais de seu sistema.
Long Date	Exibe a data de acordo com o ajuste atual de seu sistema para datas longas. Ex. Sábado, 22 de junho de 1996.
Medium Date	Exibe a data de acordo com o ajuste atual de seu sistema para datas médias. Ex. 22-Jun-96.
Short Date	Exibe a data de acordo com o ajuste atual de seu sistema para datas curtas. Ex. 22/06/96.
Long Time	Exibe a hora de acordo com o ajuste atual de seu sistema para horas longas. Ex. 20:33:51.
Medium Time	Exibe a hora de acordo com o ajuste atual de seu sistema para horas médias. Ex. 08:33 PM.
Short Time	Exibe a hora de acordo com o ajuste atual de seu sistema para horas curtas. Ex. 20:33.

## Exemplo:

```
Teste = Format(12345678, "General Number")
```

```
REM Teste = 12345678
```

```
Teste = Format(12345678, "Currency")
```

REM Teste = R\$12.345.678.00

Teste = Format(12345678, "Fixed")

REM Teste = 12345678.00

Teste = Format(12345678, "Standard")

REM Teste = 12.345.678.00

Teste = Format(12345678, "Scientific")

REM Teste = 1.23E+07

Teste = Format(12345678, "Yes/No")

REM Teste = Yes

Teste = Format(#22/6/96#, "Long Date")

REM Teste = Sábado, 22 de Junho de 1996

Teste = Format(#22/6/96#, "Medium Date")

REM Teste = 22-Jun-96

Teste = Format(#22/6/96#, "Short Date")

REM Teste = 22/06/96

Teste = Format("20:33", "Long Time")

REM Teste = 20:33:00

Teste = Format("20:33", "Medium Time")

REM Teste = 08:33 PM

Teste = Format("20:33", "Short Time")

REM Teste = 20:33

---

## FV

---

**Categoria:** Funções Financeiras

**Finalidade:** Calcula o Valor Futuro de uma anuidade ou prestação.

**Sintaxe:** FV(Rate, Nper, Pmt[, Pv[, Type]])

**Em que:**

- Rate - Taxa de juros no período.
- Nper - Número total de pagamentos.
- Pmt - Pagamento a ser feito em cada período.
- Pv - Valor presente de uma série de pagamentos.
- Type - Número indicando quando os pagamentos serão feitos. Use 0 se os pagamentos ocorrem no fim do período e 1 se os pagamentos serão feitos antecipadamente. Se omitido, 0 é assumido.

### Exemplo:

Rem Quanto obterei se depositar \$100 por seis meses a  
REM uma taxa de 2% ao mês?  
Teste = FV(0.02, 6, -100)                    `Teste = 630.81

Rem Resposta: \$ 630.81

---

### GetAttr

---

**Categoria:** Funções de Acesso a Arquivos

**Finalidade:** Retorna um número que representa os atributos de um arquivo ou diretório.

**Sintaxe:** GetAttr(Pathname)

**Em que:**

- Pathname - Expressão string que especifica o nome do arquivo. Essa string também pode conter o caminho completo (drive e diretório) do arquivo procurado. Se o caminho não for encontrado, Null será retornado.

**Valores de Retorno:**

Valor	Descrição
0	Normal
1	Read-Only (Somente Leitura)
2	Hidden (Oculto)
4	System (Sistema)
16	Diretório
32	O arquivo foi alterado após o último backup

### Exemplo:

```
Teste = GetAttr("C:\IO.SYS")                    `Teste = 7  
Rem O Arquivo IO.SYS é Read-only, Hidden e System ao mesmo tempo  
Rem por isso 1 + 2 + 4 = 7
```

```
Teste = GetAttr("C:\WINDOWS")                `Teste = 16
```

---

### Hex

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Retorna uma string representando o valor hexadecimal de um número.

**Sintaxe:** Hex(Número)

**Em que:**

- Número - Qualquer expressão numérica válida.

**Exemplo:**

```
Teste = Hex(15)           `Teste = F
Teste = Hex(5)           `Teste = 5
Teste = Hex(255)        `Teste = FF
```

---

## Hour

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o número da hora (de 0 a 23) de uma hora especificada.

**Sintaxe:** Hour(Time)

**Em que:**

- Time - Qualquer expressão numérica ou string que possa representar um intervalo de tempo. Se Time contém Null, o valor de retorno da função será Null também.

**Exemplo:**

```
Teste = Hour(#8:33:17 PM#)      `Teste = 20
```

---

## IIf

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna uma de duas partes, dependendo da avaliação de uma expressão.

**Sintaxe:** IIf(Expressão, Truepart, Falsepart)

**Em que:**

- Expressão - Expressão a ser avaliada.
- Truepart - Valor ou expressão retornada se a expressão avaliada for verdadeira.
- Falsepart - Valor ou expressão retornada se a expressão avaliada for falsa.

**Exemplo:**

```
Dado = 100
Teste = IIf(Dado > 10, "Maior que Dez", "Menor que Dez")
Rem Teste = "Maior que Dez"
```

---

## InputBox

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Exibe uma caixa de diálogo para que o usuário possa digitar uma resposta qualquer. Após escolher um dos botões disponíveis, o VB retorna o valor digitado.

**Sintaxe:** InputBox(Prompt[, Title][, Default][, Xpos][, Ypos])

**Em que:**

- Prompt - Expressão string que será exibida como mensagem na caixa de diálogo. O comprimento máximo da mensagem é de 1024 caracteres. Você pode quebrar a mensagem em várias linhas utilizando a combinação de caracteres de retorno de carro Chr(13) e mudança de linha Chr(10), como em (Chr(13) & Chr(10)).
- Title - Expressão string que será exibida na barra de título da caixa de diálogo. Se você omitir este argumento, o nome exibido na barra de título será o nome de sua aplicação.
- Default - Expressão string que será retornada como default caso o usuário não digite nenhum dado na caixa de diálogo. Se você omitir este argumento, a textbox da caixa de diálogo será exibida vazia.
- Xpos - Expressão numérica que especifica em twips a distância horizontal entre a borda esquerda da tela e da caixa de diálogo. Se você omitir este argumento, a caixa de diálogo será centralizada horizontalmente.
- Ypos - Expressão numérica que especifica em twips a distância vertical entre a borda superior da caixa de diálogo e o topo da tela. Se você omitir este argumento, a caixa de diálogo será centralizada verticalmente.

---

## InStr

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna a posição da primeira ocorrência de uma string dentro de outra.

**Sintaxe:** InStr([Início, ]String1, String2[, Compare])

**Em que:**

- Início - Expressão numérica que indica a posição inicial de cada busca. Se omitido, a busca começa desde a primeira posição.
- String1 - String em que é feita a procura.
- String2 - Expressão string que está sendo procurada.
- Compare - Especifica o tipo de comparação a ser efetuado entre as strings.

**Exemplo:**

```
Teste = InStr("Casa", "as")           `Teste = 2
Rem A string "as" foi localizada a partir da segunda posição em "Casa"
```

```
Teste = InStr("Casamento", "t")     `Teste = 8
Rem A string "t" foi localizada na oitava posição de "Casamento"
```

```
Teste = InStr("Casa", "z")           `Teste = 0
Rem A string "z" não foi localizada em "Casa"
```

---

## Int

---

**Categoria:** Funções Matemáticas

**Finalidade:** Retorna a parte inteira de um número.

**Sintaxe:** Int(Número)

**Em que:**

- Número - Qualquer expressão numérica válida. Se número contém Null, o valor de retorno será Null também.

**Exemplo:**

Esta função opera de forma idêntica à função **Fix**.

---

## IsArray

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor Boolean indicando se uma determinada variável é um Array.

**Sintaxe:** IsArray(Varname)

**Em que:**

- Varname - Qualquer variável de memória.

**Exemplo:**

```
Dim MeuArray(2) As Integer, Nome as String
Teste = IsArray(Nome)           `Teste = False
Teste = IsArray(MeuArray)      `Teste = True
```

---

## IsDate

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor Boolean indicando se uma determinada expressão pode ser convertida em uma data.

**Sintaxe:** IsDate(Expressão)

**Em que:**

- Expressão - Qualquer data ou string que possa ser reconhecida como uma data ou hora válida.

**Exemplo:**

```
Dado = "22/6/96"
Mens = "hoje"
Teste = IsDate(Dado)           `Teste = True
Teste = IsDate(Mens)          `Teste = False
```

---

## IsEmpty

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor Boolean indicando se uma determinada variável foi inicializada ou não.

**Sintaxe:** IsEmpty(Expressão)

**Em que:**

- Expressão - Qualquer expressão numérica ou string que possa ser reconhecida como um nome de variável.

**Exemplo:**

```
Teste = IsEmpty(Dado)           `Teste = True
Dado = Null
Teste = IsEmpty(Dado)           `Teste = False
```

---

## IsError

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor Boolean indicando se uma determinada expressão é um valor de erro.

**Sintaxe:** IsError(Expressão)

**Em que:**

- Expressão - Deve ser do tipo de dados Variant.

**Exemplo:**

```
MeuErro = CVErr(212)
Teste = IsError(MeuErro)      ' Teste = True

MeuErro = 212
Teste = IsError(MeuErro)      ' Teste = False
```

---

## IsMissing

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor Boolean indicando se um argumento opcional foi passado para uma procedure ou não.

**Sintaxe:** IsMissing(Nome do Argumento)

**Em que:**

- Nome do Argumento - Nome de um argumento opcional em uma procedure. A função IsMissing retorna um valor True se o argumento especificado não foi passado à função; caso contrário, a resposta será False.

**Exemplo:**

```
Function Calcula(Optional Valor)
    If IsMissing(valor) Then
        MsgBox("Não recebi o argumento valor!")
        Exit
    Else
        Valor = ((Valor / 100) + 1) ^ 1.12
```

```
        Calcula = Valor
    End If
End Function
```

---

## IsNull

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor Boolean indicando se uma expressão contém dados inválidos (Null).

**Sintaxe:** IsNull(Expressão)

**Em que:**

- Expressão - Qualquer valor numérico ou expressão string. Se a expressão avaliada tiver dados inválidos (Null), a função resultará em True; caso contrário, o valor retornado será False.

**Exemplo:**

```
Dado = ""
Teste = IsNull(Dado)           'Teste = False

Dado = Null
Teste = IsNull(Dado)          'Teste = True.
```

---

## IsNumeric

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor Boolean indicando se uma determinada expressão pode ser avaliada como um valor numérico.

**Sintaxe:** IsNumeric(Expressão)

**Em que:**

- Expressão - Qualquer valor numérico ou expressão string.

**Exemplo:**

```
Dado = "100"
Teste = IsNumeric(Dado)       'Teste = True

Dado = "cem "
Teste = IsNumeric(Dado)       'Teste = False
```

---

## LBound

---

**Categoria:** Funções de Array

**Finalidade:** Retorna o menor elemento da dimensão especificada de um determinado Array.

**Sintaxe:** LBound(Arrayname[, Dimensão])

**Em que:**

- Arrayname - Nome da variável array.

- Dimensão - Número que indica que dimensão do array está sendo inspecionada. Utilize 1 para a primeira dimensão, 2 para a segunda e assim por diante.

**Exemplo:**

```
Dim Dado(5 To 10)
Teste = LBound(Dado, 1)           `Teste = 5
```

---

**LCase**

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma string convertida em letras minúsculas.

**Sintaxe:** LCase(String)

**Em que:**

- String - Qualquer expressão string válida.

Somente as letras maiúsculas serão convertidas em minúsculas. Os demais caracteres, como letras minúsculas, números e sinais, permanecem inalterados.

**Exemplo:**

```
Dado = "CAIXA ALTA"
Teste = LCase(Dado)           `Teste = "caixa alta"
```

---

**Left**

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna o número especificado de caracteres a partir do lado esquerdo de uma string.

**Sintaxe:** Left(String, Comprimento)

**Em que:**

- String - Expressão string da qual os caracteres serão retornados.  
Comprimento - Expressão numérica indicando quantos caracteres serão retornados. Se igual a 0, uma string de tamanho zero será retornada. Se o comprimento for maior que o número de caracteres da string, toda ela será retornada.

**Exemplo:**

```
Dado = "Visual Basic 4.0"
Teste = Left(Dado, 1)           `Teste = "V"
Teste = Left(Dado, 6)           `Teste = "Visual"
Teste = Left(Dado, 10)          `Teste = "Visual Bas"
```

---

**Len**

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna o número de caracteres em uma string ou o número de bytes requeridos para armazenar uma variável.

**Sintaxe:** Len(String | Varname)

**Em que:**

- String - Qualquer expressão string válida. Se a string contém Null, Null é retornado.
- Varname - Qualquer nome de variável válido. Se Varname contém Null, Null é retornado. Se Varname é do tipo de dados Variant, Len trata a variável como uma string, retornando o número de caracteres que ela contém.

**Exemplo:**

```
Dado = "Estudo de Len"  
Teste = Len (Dado)           `Teste = 13  
Teste = Len("Visual")       `Teste = 6  
Teste = Len("Visual Basic") `Teste = 12
```

---

## LoadPicture

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Carrega uma imagem dentro de um objeto Form object, Controle PictureBox ou controle Image.

**Sintaxe:** LoadPicture([Stringexpression])

**Em que:**

- Stringexpression - Nome do arquivo gráfico a ser carregado.

Os formatos de arquivo suportados pelo Visual Basic são os seguintes: Bitmap (.BMP), Icon (.ICO), Run-Length Encoded (.RLE) e Metafile (.WMF).

**Exemplo:**

```
Rem Para exibir a imagem como fundo de um Form, PictureBox ou controle  
Rem Image atribua a figura à propriedade Picture.  
Form1.Picture = LoadPicture("PARTY.BMP")
```

```
Rem Para atribuir um ícone a um formulário ajuste a propriedade Icon:  
Form1.Icon = LoadPicture("PHONE01.ICO")
```

---

## Log

---

**Categoria:** Funções Matemáticas

**Finalidade:** Calcula o logaritmo natural de um número.

**Sintaxe:** Log(Número)

**Em que:**

- Número - Qualquer expressão numérica válida maior que zero.

**Exemplo:**

```
Teste = Log(0)           `Teste = 1  
Teste = Log(5)          `Teste = 1.6094379124341
```

---

## LTrim

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma cópia de uma string eliminando os espaços em branco existentes entre o início da string e o primeiro caractere.

**Sintaxe:** LTrim(String)

**Em que:**

- String - Qualquer expressão string válida. Se a string contém Null, Null é retornado.

**Exemplo:**

```
Dado = " Casa"
Teste = LTrim(Dado)      `Teste = "Casa"
```

---

## Mid

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna o número especificado de caracteres de uma string.

**Sintaxe:** Mid(String, Start[, Length])

**Em que:**

- String - Qualquer expressão string válida. Se a string contém Null, Null é retornado.
- Start - Posição inicial a partir da qual os caracteres serão selecionados. Se a posição inicial for maior que o número de caracteres da string, uma string de tamanho zero será retornada.
- Length - Número de caracteres que serão selecionados. Se omitido, todos os caracteres desde a posição inicial especificada até o fim dela serão selecionados.

**Exemplo:**

```
Dado = "Visual Basic"
Teste = Mid(Dado,1, 3)      `Teste = "Vis"
Teste = Mid(Dado,4) `Teste = "ual Basic"
Teste = Mid(Dado,5,5)      `Teste = "al Ba"
```

---

## Minute

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o número de minutos da hora especificada (de 0 a 59)

**Sintaxe:** Minute(Time)

**Em que:**

- Time - Qualquer expressão numérica ou string que possa representar um intervalo de tempo. Se Time contém Null, o valor de retorno da função será Null também.

## Exemplo:

```
Teste = Minute(#8:33:17 PM#)
```

```
`Teste = 33
```

---

## Month

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o número do mês de uma determinada data (de 1 a 12).

**Sintaxe:** Month(Data)

**Em que:**

- Data - A data a ser avaliada pela função.

## Exemplo:

```
Teste = Month(#22/06/96#)
```

```
`Teste = 6
```

---

## MsgBox

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Exibe uma mensagem em um quadro de diálogo e espera que o usuário escolha um dos botões disponíveis. Após a seleção, o VB retorna um valor indicando o botão escolhido.

**Sintaxe:** MsgBox(Prompt[, Buttons][, Title])

**Em que:**

- Prompt - Expressão string que será exibida como mensagem na caixa de diálogo. O comprimento máximo da mensagem é de 1024 caracteres. Você pode quebrar a mensagem em várias linhas, utilizando a combinação de caracteres de retorno de carro Chr(13) e mudança de linha Chr(10), como em (Chr(13) & Chr(10)).
- Buttons - Expressão numérica que é a soma dos valores que especificam o número e tipo de botões a serem exibidos. Você também pode especificar o estilo de ícone a ser usado, o botão default e a modalidade da caixa de mensagem. Se omitido, o valor default para os botões é zero.
- Title - Expressão string que será exibida na barra de título da caixa de diálogo. Se você omitir este argumento, o nome exibido na barra de título será o nome de sua aplicação.

## Ajustes:

Aparência:

Ajuste	Descrição
0	Exibe somente o botão OK (default)
1	Exibe botões OK e Cancel
2	Exibe botões Abort, Retry e Ignore
3	Exibe Yes, No e Cancel
4	Exibe os botões Yes e No

5	Exibe Retry e Cancel
16	Exibe ícone de Parada Crítica
32	Exibe ícone de Pergunta
48	Exibe ícone de Atenção
64	Exibe ícone de Informação
0	O primeiro botão é default
256	O segundo botão é default
512	O terceiro botão é default
0	Aplicação modal. O usuário deve responder à mensagem exibida antes de prosseguir com a execução da aplicação atual.
4096	Sistema modal. Todas as aplicações são suspensas até o usuário responder à message box.

O primeiro grupo de valores descreve o número e o tipo de botões exibidos. O segundo grupo (16, 32, 48, 64) descreve o estilo do ícone exibido. O terceiro grupo (0, 256, 512) determina que botão é o default. O quarto e último grupo (0, 4096) determina a modalidade da caixa de diálogo. Você deve selecionar um número de cada grupo e somar todos para obter a aparência desejada para sua Message Box.

Valores de Retorno:

Valor	Botão Escolhido
1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

### Exemplo:

Rem Exibe botões OK e Cancel e ícone de Parada Crítica  
 Teste = MsgBox("Teste", 17, "MsgBox")

Rem Exibe Yes/No numa janela de Atenção com o primeiro botão default  
 Teste = MsgBox("Teste", 68, "MsgBox")



---

## RGB

---

**Categoria:** Funções de Sistema Operacional

**Finalidade:** Retorna um número representando uma cor do padrão RGB.

**Sintaxe:** RGB (Red, Green, Blue)

**Em que:**

- Red - Número entre 0 to 255, inclusive, que representa o componente vermelho de uma cor.
- Green - Número entre 0 to 255, inclusive, que representa o componente verde de uma cor.
- Blue - Número entre 0 to 255, inclusive, que representa o componente azul de uma cor.

**Exemplo:**

```
Commdialog.Color = RGB(255, 0, 0)
```

---

## Right

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna o número especificado de caracteres a partir do lado direito de uma string.

**Sintaxe:** Right(String, Comprimento)

**Em que:**

- String - Expressão string da qual os caracteres serão retornados.
- Comprimento - Expressão numérica indicando quantos caracteres serão retornados. Se igual a 0, uma string de tamanho zero será retornada. Se o comprimento for maior que o número de caracteres da string, toda ela será retornada.

**Exemplo:**

```
Dado = "Visual Basic 4.0"  
Teste = Right(Dado, 1)           `Teste = "0"  
Teste = Right(Dado, 6)         `Teste = "ic 4.0"  
Teste = Right(Dado, 10)        `Teste = " Basic 4.0"
```

---

## Rnd

---

**Categoria:** Funções Matemáticas

**Finalidade:** Retorna um número aleatório (randômico).

**Sintaxe:** Rnd[(Número)]

**Em que:**

- Número - Qualquer expressão numérica válida.

Esta função retorna um valor menor que 1 e maior ou igual a zero.

**Exemplo:**

```
Teste = Rnd
```

---

**Rtrim**

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma cópia de uma string, eliminando os espaços existentes entre o último caractere da string e o fim dela.

**Sintaxe:** RTrim(String)

**Em que:**

- String - Qualquer expressão string válida. Se a string contém Null, Null é retornado.

**Exemplo:**

```
Dado = "Casa      "  
Teste = LTrim(Dado)      `Teste = "Casa"
```

---

**Second**

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o número de segundos da hora especificada (de 0 a 59).

**Sintaxe:** Second(Time)

**Em que:**

- Time - Qualquer expressão numérica ou string que possa representar um intervalo de tempo. Se Time contém Null, o valor de retorno da função será Null também.

**Exemplo:**

```
Teste = Second(#8:33:17 PM#)      `Teste = 17
```

---

**Sgn**

---

**Categoria:** Funções Matemáticas

**Finalidade:** Retorna um inteiro conforme o sinal do número.

**Sintaxe:** Sgn(Número)

**Em que:**

- Número - Qualquer expressão numérica válida.

**Ajustes:**

Se Número:	Sgn Retorna:
Maior que zero	1
Igual a zero	0
menor que zero	-1

**Exemplo:**

```
Teste = Sgn (12)      ' Teste = 1
Teste = Sgn (-2.4)   ' Teste = -1
Teste = Sgn (0)      ' Teste = 0
```

---

**Shell**

---

**Categoria:** Funções de Sistema Operacional**Finalidade:** Inicia a execução de um programa qualquer.**Sintaxe:** Shell(Pathname[, Windowstyle])**Em que:**

- Pathname - Nome do programa a ser executado. Pode incluir eventuais argumentos na linha de comando. Também pode incluir o diretório e drive onde está localizado o executável. Você também pode informar o nome do documento se ele tiver sua extensão associada a um executável no Windows (exemplo: .TXT abre o Notepad).
- Windowstyle - Número que corresponde ao estilo da janela em que o programa será executado. Se omitido, o programa será executado minimizado.

**Ajustes:**

Valor	Descrição
0	A janela é oculta e o foco é passado para essa janela.
1	A janela tem o foco e é restaurada para seu tamanho e posição originais.
2	A janela é exibida como um ícone com foco.
3	A janela é maximizada com foco.
4	A janela é restaurada ao seu mais recente tamanho e posição. A janela corrente permanece ativa.
6	A janela é exibida como um ícone. A janela corrente permanece ativa.

**Exemplo:**

```
Teste = Shell ("C:\WINDOWS\notepad.exe", 1)
```

---

**Sin**

---

**Categoria:** Funções Matemáticas**Finalidade:** Calcula o seno de um número.**Sintaxe:** Sin(Número)**Em que:**

- Número - Qualquer expressão numérica válida que expresse um ângulo em radianos.

**Exemplo:**

```
Teste = Sin(1.25)           \Teste = 0.948984619355586
Teste = Sin(.79)           \Teste = 0.710353272417608
```

**Space**

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma string com o número de espaços em branco especificado pelo usuário.

**Sintaxe:** Space(Número)

**Em que:**

- Número - A quantidade de espaços em branco que você quer incluir na string.

**Exemplo:**

```
Dado = "Passo"
Teste = Dado & Space(2) & Dado  \Teste = "Passo  Passo"
```

**Sqr**

**Categoria:** Funções Matemáticas

**Finalidade:** Calcula a raiz quadrada de um número.

**Sintaxe:** Sqr(Número)

**Em que:**

- Número - Qualquer expressão numérica válida maior ou igual a zero.

**Exemplo:**

```
Teste = Sqr(16)           \Teste = 4
```

**Str**

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Retorna uma string representando o valor de um número.

**Sintaxe:** Str(Número)

**Em que:**

- Número - Qualquer expressão numérica válida.

**Exemplo:**

```
Teste = Str(123)           ' Teste = " 123"
Teste = Str(2000)         ' Teste = "2000"
```

**StrComp**

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna um valor indicando o resultado de uma comparação de strings (se maior, igual ou menor).

**Sintaxe:** StrComp(String1, String2[, Compare])

**Em que:**

- String1 - Qualquer expressão string válida.
- String2 - Qualquer expressão string válida.
- Compare - Especifica o tipo de comparação a ser efetuado.

**Ajustes:**

Se:	StrComp Retorna:
String1 é Menor que String2	-1
String1 é igual à String2	0
String1 é maior que String2	1
String1 ou String2 é Null	Null

**Exemplo:**

```
Dado1 = "ABCD"
```

```
Dado2 = "abcd"
```

```
Teste = StrComp(Dado1, Dado2, 1) ' Teste = 0.
```

```
Teste = StrComp(Dado1, Dado2, 0) ' Teste = -1.
```

```
Teste = StrComp(Dado2, Dado1) ' Teste = 1.
```

---

## StrConv

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma string convertida de acordo com a operação especificada pelo usuário.

**Sintaxe:** StrConv(String, Conversion)

**Em que:**

- String - A string a ser convertida.
- Conversion - A soma dos valores que especificam o tipo da conversão a ser feita.

**Ajustes:**

Valor	Descrição
1	Converte a string em caracteres maiúsculos.
2	Converte a string em caracteres minúsculos.
3	Converte a primeira letra de cada palavra da string em maiúscula.

**Exemplo:**

```
Dado = "visual basic"
```

```
Teste = StrConv(dado, 3)
```

```
'Teste = "Visual Basic"
```

---

## String

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma string com caracteres repetidos no comprimento especificado pelo usuário.

**Sintaxe:** String(Número, Caractere)

**Em que:**

- Número - Comprimento da string a ser retornada.
- Caractere - Código do caractere ou expressão string cujo primeiro caractere será utilizado para construção da nova string.

**Exemplo:**

```
Teste = String(5, "*")           `Teste = "*****"  
Teste = String(5, 42)           `Teste = "*****".  
Teste = String(10, "ABC") `Teste = "AAAAAAAAAA".
```

---

## Tan

---

**Categoria:** Funções Matemáticas

**Finalidade:** Calcula a tangente de um número.

**Sintaxe:** Tan(Número)

**Em que:**

- Número - Qualquer expressão numérica válida que expresse um ângulo em radianos.

**Exemplo:**

```
Teste = Tan(.79)                 `Teste = 1.00924628838275
```

---

## Time

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna a hora atual do sistema.

**Sintaxe:** Time

**Exemplo:**

```
Teste = Time                     `Teste = 23:44:07
```

---

## Timer

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o número de segundos decorridos desde à 0:00h (de 0 a 59).

**Sintaxe:** Timer

**Exemplo:**

```
Teste = Timer                     `Teste = 85591.6
```

---

## TimeSerial

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna um Variant com o valor da hora, minuto e segundo especificado.

**Sintaxe:** TimeSerial(Hora, Minuto, Segundo)

**Em que:**

- Hora - Número entre 0 e 23 inclusive.
- Minuto - Qualquer expressão numérica.
- Segundo - Qualquer expressão numérica.

**Exemplo:**

```
REM Retorna a hh:mm:ss de 20:13:45
Teste = TimeSerial(20, 13, 45)           `Teste = 20:13:45
```

---

## TimeValue

---

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna um valor de hora.

**Sintaxe:** TimeValue(Tempo)

**Em que:**

- Tempo - Expressão string representando uma hora entre 0:00:00 e 23:59:59 inclusive. Você também pode utilizar qualquer expressão que represente uma indicação de tempo dentro da faixa de valores válidos. O VB também aceita a notação AM/PM como em 2:00PM.

**Exemplo:**

```
REM Converte uma string num horário
Teste = TimeValue("14:00:23")           `Teste = 14:00:23
```

---

## Trim

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma cópia de uma string, eliminando os espaços existentes entre o início da string e o primeiro caractere, e o último caractere da string e o fim dela.

**Sintaxe:** Trim(String)

**Em que:**

- String - Qualquer expressão string válida. Se a string contém Null, Null é retornado.

**Exemplo:**

```
Dado = "      Casa      "
Teste = LTrim(Dado)           `Teste = "Casa"
```

---

## TypeName

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna uma string que informa qual é o tipo de dados da variável analisada.

**Sintaxe:** TypeName(Varname)

**Em que:**

Varname - Qualquer variável de memória.

**Ajustes:**

String Retornada:	Variável Contém:
Byte	Byte
Integer	Integer
Long	Long integer
Single	Single
Double	Double
Currency	Currency
Date	Date
String	String
Boolean	Boolean
Error	Um valor de erro
Empty	Não inicializada
Null	Dado Inválido
Object	Um objeto que suporta OLE

**Exemplo:**

```
Dim Dado As Integer
```

```
Teste = TypeName(Dado)
```

```
Teste = "Integer"
```

---

## Ubound

---

**Categoria:** Funções de Array

**Finalidade:** Retorna o maior elemento da dimensão especificada de um determinado array.

**Sintaxe:** UBound(Arrayname[, Dimensão])

**Em que:**

- Arrayname - Nome da variável array.
- Dimensão - Número que indica que dimensão do array está sendo inspecionada. Utilize 1 para a primeira dimensão, 2 para a segunda e assim por diante.

**Exemplo:**

```
Dim Dado(5 To 10)
```

Teste = UBound(Dado, 1)

'Teste = 10

---

## UCase

---

**Categoria:** Funções de Manipulação de Strings

**Finalidade:** Retorna uma string convertida em letras maiúsculas.

**Sintaxe:** UCase(String)

**Em que:**

- String - Qualquer expressão string válida.

Somente as letras minúsculas serão convertidas em maiúsculas. Os demais caracteres, como números e sinais, permanecem inalterados.

**Exemplo:**

Dado = "caixa alta"

Teste = UCase(Dado)                    'Teste = "CAIXA ALTA"

---

## Val

---

**Categoria:** Funções de Conversão de Dados

**Finalidade:** Retorna os números contidos em uma string.

**Sintaxe:** Val(String)

**Em que:**

- String - Qualquer expressão string válida.

**Exemplo:**

Teste = Val("2457")                    'Teste = 2457.

Teste = Val(" 2 45 7")                'Teste = 2457.

Teste = Val("24 and 57")              'Teste = 24.

---

## VarType

---

**Categoria:** Funções de Verificação

**Finalidade:** Retorna um valor indicando o subtipo de uma variável.

**Sintaxe:** VarType(Varname)

**Em que:**

- Varname - Qualquer variável de memória válida.

**Ajustes:**

Valor	Descrição da Variável
0	Empty
1	Null
2	Integer
3	Long integer
4	Single
5	Double

6	Currency
7	Date
8	String
9	OLE Automation object
10	Error
11	Boolean
12	Variant (usada somente com arrays de Variants)
13	Não OLE Automation object
17	Byte
8192	Array

### Exemplo:

```
Dim Dado As Integer
Teste = VarType(Dado)           `Teste = 2
```

### Weekday

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o número do dia da semana (1 - Domingo, etc.).

**Sintaxe:** Weekday(Data, [Firstdayofweek])

**Em que:**

- Data - Expressão string ou numérica representando uma data. Se Data contém Null, Null é retornado.
- Firstdayofweek - Uma constante que define o primeiro dia da semana. Caso você não especifique, o domingo será considerado como primeiro dia. Veja a seção Ajustes da função DateDiff para maiores detalhes.

### Exemplo:

```
Teste = WeekDay(#6/22/96#)     `Teste = 7
```

### Year

**Categoria:** Funções de Data e Hora

**Finalidade:** Retorna o número do ano de uma determinada data.

**Sintaxe:** Year(Data)

**Em que:**

- Data - Expressão string ou numérica representando uma data. Se Data contém Null, Null é retornado.

### Exemplo:

```
Teste = Year(#6/22/96#)       `Teste = 1996
```