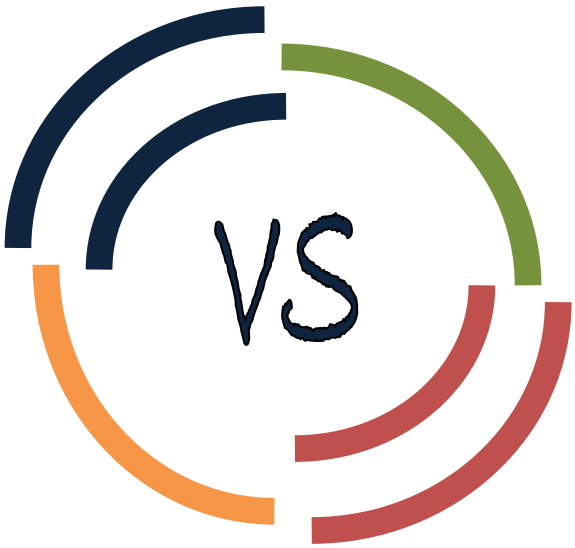
Tutorial VBScript



Vincenzo Storace

ver 1.0

INTRODUZIONE

Per l'utente che ha imparato a creare pagine in HTML, ad un certo punto nasce l'esigenza di pretendere di più dal proprio lavoro. Occorre allora imparare un linguaggio di **scripting**. Vedremo che esistono:

- **script lato client**, che sono eseguiti direttamente dal browser senza il bisogno di ricaricare la pagina
- **script lato server**, che sono eseguiti su server e che ad ogni richiesta generano una nuova pagina da inviare al browser.

Con VbScript si possono realizzare sia script lato server (con ASP) sia script lato client (in genere supportati solo da IE).
I linguaggi di scripting si inseriscono dentro il codice HTML per rendere una pagina dinamica e interattiva con l'utente; ad esempio, uno di questi linguaggi è ASP che in particolare è un linguaggio per script lato server.
Con ASP in realtà si può usare anche Javascript, ma è uso comune programmare le pagine ASP con VbScript.

AGGIUNTA DI CODICE VBSCRIPT IN UNA PAGINA

In HTML esiste il tag<script> che viene usato per includere il codice di un linguaggio di scripting. Nel caso del VBScript la sintassi è questa:

```
<script language="VBScript">
<!–
……CODICE VBscript…..
–>
</script>
```

Quando la pagina HTML arriva al browser, questo interpreta il codice compreso nel tag<script>. Attraverso l'attributo Language viene indicato il linguaggio di scripting utilizzato. Il codice VBScript è inserito tra il tag commento <!– –>, in questo modo il codice non viene visualizzato nei browser che non supportano il tag<script>.
Dove inserire questo codice? Dipende dalla sua funzione. Può essere inserito nel HEAD o nel BODY. Uno script che visualizza un messaggio, sicuramente va inserito nel BODY.
Passiamo ad un esempio pratico. Cominciamo con una pagina che visualizza il classico massaggio "Ciao Mondo". Occorre conoscere il comando che permette di visualizzare un messaggio: **document.write**.

```
<html>
<head>
<title>esempio 1</title>
</head>
<body>
<script language="VBScript">
<!–
 document.write "Ciao Mondo"
//–>
</script>
</body>
</html>
```

VARIABILI

Il VBScript è un derivato del Visual Basic, e le variablili si comportano quasi nello stesso modo. Per dichiararle si usa l'istruzione **Dim.** Ad esempio
Dim variabile1
Dim variabile2
ecc.
Oppure
Dim variabile1, variabile2, ecc.
Chi è abituato ad altri linguaggi sa che dichiarando una variabile bisogna anche definire il tipo, cioè se conterrà un numero, una stringa, una data ecc.
In VBScript questa distinzione non c'è: non è necessario dichiarare il tipo delle variabili. La gestione di un numero o di una stringa viene affidata all'interprete VBScript. Questo rende molto più semplice programmare, ma bisogna stare attenti, come nel prossimo esempio:

```
<script language="VBScript">
<!–
 dima,b,c
 a=inputbox("Primo Numero:")
 b=inputbox("Secondo Numero:")
 c=a+b
 document.write c
//–>
</script>
```

Nell'esempio, assegno i valori alle variabili a e b attraverso le **InputBox**, cioè finestre di dialogo che chiedono all'utente di inserire un valore. Se assegno il valore 3 la prima volta e 5 la seconda, ci si aspetta che la riga "document.write c" stampi il valore 8 ed invece visualizza 35.
Questo perché a e b vengono interpretate come stringhe, e il segno + invece di fare la somma concatena le 2 stringhe. Per risolvere il problema bisogna convertire le variabili da stringa a numerico.
Attenzione: i tipi non vanno dichiarati ma esistono e sono:

- **Boolean**
  Può contenere True o False.
- **Byte**
  Contiene un intero compreso tra 0 e 255.
- **Integer**
  Contiene un intero compreso tra -32.768 e 32.767.
- **Currency**
  Contiene un valore compreso tra -922.337.203.685.477,5808 e 922.337.203.685.477,5807.
- **Long**
  Contiene un intero compreso tra -2.147.483.648 e 2.147.483.647.
- **Single**
  Contiene un numero in virgola mobile e precisione singola compreso tra -3,402823E38 e 3,402823E38
- **Double**
  Contiene un numero in virgola mobile e precisione doppia compreso tra -1,79769313486232E308 e 1,79769313486232E308
- **Date**
  Contiene un numero che rappresenta una data compresa tra l'1 gennaio dell'anno 100 e il 31 dicembre del 9999.

- **String**
  Contiene una stringa di lunghezza variabile composta da un massimo di circa 2 miliardi di caratteri.
- **Object**
  Contiene un oggetto.

La conversione dei tipi viene gestita automaticamente dal VBScript quando è chiara. Nell'esempio precedente, se si facesse la sottrazione il risultato sarebbe corretto:

```
<script language="VBScript">
<!–
dim a,b,c
a=inputbox("Primo Numero:")
b=inputbox("Secondo Numero:")
c=a-b
document.write c
//–>
</script>
```

Nel primo esempio, si ha il + che è utilizzato sia per la somma che per la concatenazione. Quindi a e b non vengono convertite in tipo numerico, ma sono lasciate in tipo stringhe. Per forzare questa conversione si utilizza la funzione **Cint:**

```
<script language="VBScript">
<!–
 dima,b,c
 a=inputbox("Primo Numero:")
 b=inputbox("Secondo Numero:")
 c=Cint(a)+Cint(b)
document.write c
//–>
</script>
```

L'istruzione **dim** serve a dichiarare una variabile, ma questa dichiarazione non è obbligatoria: il programma funzionerebbe anche senza la riga "dima,b,c". Ciò rende la programmazione ancora più veloce perché non bisogna dichiarare le variabili, quando occorrono basta definirle.
Quando si realizza un programma complicato è consigliabile usare il comando "Option Explicit" all'inizio del programma Questo comando obbliga la dichiarazione delle variabili ed è utile se si sbaglia a scrivere il nome di una variabile. Infatti, se sbaglio il nome questa non esiste e si ha un errore.

```
<script language="VBScript">
<!–
 Option Explicit
 dima,b,c
 a = inputbox("Primo Numero:")
 b = inputbox("Secondo Numero:")
 c = Cint(a)+Cint(b)
 document.write c
//–>
</script>
```

Quando si sceglie il **nome di una variabile** bisogna tener presenti le seguenti regole:

- Deve iniziare con una lettera dell'alfabeto.
- Non può includere punti e spazi e il trattino meno.
- Non deve essere composta da più di 255 caratteri.
- Non c'è differenza tra lettere maiuscole e minuscole.

C'è poi la regola non obbligatoria di dare dei **nomi logici** alle variabili e così una variabile che deve contenere la data di nascita si potrà chiamare dtData_Di_Nascita, dove dt indica che è di tipo data. Questo permette di realizzare programmi più leggibili.

OPERATORI

Gli operatori in VBScript si possono suddividere in tipi:

Operatori Aritmetici

		a=5 b=2	Risultato
+	Somma	c=a+b	7
-	Sottrazione	c=a-b	3
*	Moltiplicazione	c=a*b	10
/	Divisione	c=a/b	2,5
	Divisione Intera	c=ab	2
Mod	Modulo	c=a Mod b	1
^	Elevamento a potenza	c=a^b	25
&	Concatenamento di strighe	c=a & b	52

Per le formule complesse hanno precedenza addizione e sottrazione da sinistra verso destra.

Operatori di confronto

=	Uguaglianza
>=	Maggiore o uguale a
<=	Minore o uguale a
<>	Diverso
<	Minore
>	Maggiore

Il confronto tra due variabili restituisce Vero o Falso

Operatori Logici

Not	Negazione
And	Congiunzione Logica
Or	Disgiunzione Logica

STRUTTURE CONDIZIONALI

Vediamo ora di inserire qualche bivio nei programmi Per eseguire delle operazioni sotto certe **condizioni** il comando principale è IF. La sintassi è la seguente:

```
If<condizione>Then
        <operazioni se la condizione e vera>
Else
        <operazioni se la condizione e falsa>
End If
```

Nel seguente esempio si usa la funzione Month(), che restituisce il mese in formato numerico, e Now(), che restituisce la data e l'ora attuale:

```
<script language="VBScript">
<!–
```

```
if month(Now())=6 then
        document.write “È giugno”
end if
//–>
</script>
```

Se questo è il mese di giugno, stamperà “è giugno”.
Per rendere più completo il programma:

```
<script language=”VBScript”>
<!–
if month(Now())=6 then
        document.write “É giugno”
else
         document.write “Non è giugno”
end if
//–>
</script>
```

Si possono anche creare delle strutture dentro altre strutture ossia delle **strutture annidate**:

```
<script language=”VBScript”>
<!–
if month(Now())<=6 then
        document.write “Siamo nel primo semestre dell’anno <br>”
        if month(Now())<=3 then
                document.write “e nel primo trimestre.”
        else
                document.write “e nel secondo trimestre.”
        end if
else
        document.write “Siamo nel secondo semestre dell’anno <br>”
        if month(Now())<=9 then
                document.write “e nel terzo trimestre.”
        else
                document.write “e nel quarto trimestre.”
        end if
end if
//–>
</script>
```

Si noti l’importanza di una formattazione di tipo “indent” (ossia con le rientranze) quando si usano strutture così complesse.
Quando si utilizzano strutture con molte scelte dipendenti dal valore di un parametro, si può utilizzare la struttura con SELECT.
Ecco lo stesso programma di prima con la struttura SELECT:

```
<script language=”VBScript”>
<!–
Select case month(Now())
```

```
case 1,2,3
        document.write “Siamo nel primo trimestre.”
case 4,5,6
        document.write “Siamo nel secondo trimestre.”
case 7,8,9
        document.write “Siamo nel terzo trimestre.”
case else
        document.write “Siamo nel quarto trimestre.”
end select
//–>
</script>
```

In questa struttura, viene definita la variabile da controllare con l’istruzione “Select case *Variabile*“, e vengono poi elencati i valori possibili che può assumere la variabile con l’istruzione “case *valore1*,*valore2*,*valore3*“; seguono a questa istruzione le operazioni da eseguire. Nell’esempio la variabile è numerica, nel caso di variabile stringa i valori vanno messi tra virgolette:
case “*valore1*“,”*valore2*“,”*valore3*“

STRUTTURE PER CICLI

Un ciclo serve a ripetere delle operazioni per un certo numero di volte, oppure finchè una certa condizione non avviene.
Il ciclo **FOR – NEXT** incrementa ad ogni ciclo una variabile. Quando questa variabile sarà arrivata ad una valore stabilito, il ciclo finirà.

```
<script language=”VBScript”>
<!–
Option Explicit
Dim i
For i=4 to 20
        document.write i & “<br>”
next
//–>
</script>
```

Questo programma stampa i valori da 4 a 20 incolonnati.
Attraverso il parametro STEP, posso anche contare al contrario e con passi diversi:

```
<script language=”VBScript”>
<!–
Option Explicit
Dim i
For i=20 to 4 step -2
        document.write i & “<br>”
next
//–>
</script>
```

Questo programma visualizzerà in numeri 20, 18, 16, 14, 12, 10, 8, 6, 4
Simile è il ciclo **FOR – EACH**. Questo ciclo si basa su un insieme di oggetti e viene ripetuto per ogni elemento dell’insieme.

In questo esempio viene usato un vettore. Dopo essere stato dichiarato e riempito, attraverso un ciclo For Each viene visualizzato:

```
<script language=”VBScript”>
<!–
Option Explicit
Dim vettore(5),elemento
vettore(0)=”Html”
vettore(1)=”Asp”
vettore(2)=”Php”
vettore(3)=”JavaScript”
vettore(4)=”VBScript”
for each elemento in vettore
        document.write elemento & “<br>”
next
//–>
</script>
```

Il ciclo **DO LOOP** viene ripetuto finchè una condizione non diventa falsa. Il codice seguente viene ripetuto finchè non è trascorso un secondo da quando parte. Ad ogni ciclo incrementa la variabile conta. É un modo per apprezzare le qualità del proprio computer!

```
<script language=”VBScript”>
<!–
Option Explicit
dimTempoStart, TempoEnd, conta
conta=0
TempoStart=now()
TempoEnd=now()+1/(100000)
do Whilenow()<=TempoEnd
  conta=conta+1
loop
document.write “Il ciclo si è ripetuto ” & conta & ” volte”
document.write ” in 1 secondo”
//–>
</script>
```

Se la condizione non è mai verificata, il ciclo non è eseguito neanche una volta. Spostando la condizione al fondo, viene eseguito il test e poi verificata la condizione.
Nel seguente esempio la parola “Test Until” viene visualizzata anche se la condizione è falsa.

```
<script language=”VBScript”>
<!–
Option Explicit
dim a
a=10
do
  document.write “Test Until”
loop While a < 0
//–>
</script>
```

I cicli possono essere anche annidati.

FUNZIONI DATA E ORA

Ecco una carrellata di funzioni sulla data e l'ora:

Date	Restituisce la data	**MyDate = Date** l valore di MyDate è la data di sistema corrente.
time	Restituisce l'ora	**MyTime = Time** Restituisce l'ora di sistema corrente
now	Restituisce la data e l'ora	**MyVar = Now** Il valore di MyVar è la data e l'ora corrente
DateAdd	Aggiunge un intervallo di tempo ad una data	La funzione DateAdd non restituisce una data non valida. Nell'esempio seguente viene aggiunto un mese alla data 31 gennaio: **NewDate = DateAdd("m", 1, "31-Jan-95")** In questo caso la funzione restituisce 28-feb-95, non 31-feb-95. Se data h 31-gen-96, viene restituito 29-feb-96 perché il 1996 h un anno bisestile.
DateDiff	Restituisce il tempo tra due date	**DiffADate = "Giorni mancanti: " &DateDiff("d", Now, "1/1/2010")** **DiffADate = "Giorni mancanti: " &DateDiff("yyyy", Now, "1/1/2010")**
Day	Restituisce il giorno di una data	**MyDay = Day("19/10/1962")** Il valore di MyDay è 19
Month	Restituisce il mese di una data	**MyMonth = Month("19/10/1962")** Il valore di MyMonth è 10
Year	Restituisce l'anno di una data	**MyYear = Month("19/10/1962")** Il valore di MyYear è 1962
MonthName	Restituisce il nome del mese	**MyMonthName = Month("19/10/1962")** Il valore di MyMonthName è Ottobre
WeekDay	Restituisce un numero corrispondente al giorno della settimana	**MyWeekDay = Weekday("19/10/1962")** Il valore di MyWeekDay è 6 perché MyDate corrisponde a un venerdì
WeekDayName	Restituisce il nome del giorno della settimana	**MyWeekDay = WeekdayName("19/10/1962")** Il valore di MyWeekDay è venerdì
Second	Restituisce i secondi	**MySec = Second(Now)** Il valore di MySec è un numero che rappresenta i secondi
Minute	Restituisce i minuti	**MyVar = Minute(Now)** Il valore di MyVar è un numero che rappresenta i minuti
Hour	Restituisce l'ora	**MyVar = Hour(Now)** Il valore di MyVar è un numero che rappresenta l'ora in formato (0-23)
FormatDateTime	Restituisce la data e l'ora	**FormatDateTime(now, 0)**

	secondo un formato	Restituisce 08/03/2013 16.22.48 **FormatDateTime(now, 1)** Restituisce venerdì 8 marzo 2013 **FormatDateTime(now, 2)** Restituisce 08/03/2013 **FormatDateTime(now, 3)** Restituisce 16.22.48 **FormatDateTime(now, 4)** Restituisce 16.22

Ecco un piccolo programmino sulle funzioni data

```
<script language=”VBScript”>
<!–
Option Explicit
dimDataNascita
dim giorni(7)
giorni(2)=”Lunedi”
giorni(3)=”Martedi”
giorni(4)=”Mercoledi”
giorni(5)=”Giovedi”
giorni(6)=”Venerdi”
giorni(6)=”Sabato”
giorni(1)=”Domenica”
DataNascita=InputBox (“Quando sei nato?”)
document.write “Sei nato di ” & giorni(weekday(DataNascita)) & “<br>”
document.write “Da quando sei nato sono passati ” &datediff(“d”,DataNascita,now) & ” giorni<br>”
document.write “Da quando sei nato sono passati ” &datediff(“m”,DataNascita,now) & ” mesi<br>”
document.write “Da quando sei nato sono passati ” &datediff(“ww”,DataNascita,now) & ” settimane<br>”
//–>
</script>
```

FUNZIONI SULLE STRINGHE

Ecco una carrellata di funzioni sulle stringhe:

Asc	Restituisce il codice ASCII di un carattere	**MyNumber = Asc(“A”)** Restituisce 65 **MyNumber = Asc(“a”)** Restituisce 97 **MyNumber = Asc(“Albero”)** Restituisce 65
Chr	Restituisce un carattere dato il suo codice ASCII	**MyChar = Chr(65)** Restituisce A. **MyChar = Chr(97)** Restituisce a. **MyChar = Chr(62)** Restituisce >. **MyChar = Chr(37)** Restituisce %.

Instr	Restituisce la posizione di una stringa in un'altra	**SearchString ="visulabasicscript"** Stringa in cui eseguire la ricerca. **SearchChar = "i"** Esegue la ricerca della lettera"i". **MyPos = Instr(4, SearchString, SearchChar)** Confronto testuale a partire dalla posizione 4. Restituisce 10
LCase	Converte in minuscolo	**MyString = "VBSCript"** **LCaseString = LCase(MyString)** Il valore di LCaseString è "vbscript".
Left	Restituisce un numero di caratteri dalla sinistra di una stringa	**MyString = "VBSCript"** **LeftString = Left(MyString, 3)** ' Il valore di LeftString è "VBS".
Len	Restituisce il numero di caratteri di una stringa	**MyString = Len("VBSCRIPT")** ' MyString include 8 caratteri
LTrim	Toglie gli spazi a sinistra	**MyVar = LTrim(" vbscript ")** Il valore di MyVar è "vbscript
Mid	Restituisce un certo numero di caratteri da una stringa	**MyVar = Mid("VB Script è divertente!", 4, 6)** Il valore di MyVar è "Script".
Replace	Restituisce una data sottostringa con un'altra	**MyString = Replace("Linguaggio JavaScript ", "Java", "VB")** Restituisce "VBScript".
Rigth	Restituisce un numero di caratteri dalla destra di una stringa	**AnyString = "Salve gente!"** **MyStr = Right(AnyString, 1)** Restituisce "e". **MyStr = Right(AnyString, 6)** Restituisce " gente". **MyStr = Right(AnyString, 20)** Restituisce "Salve gente".
Rtrim	Toglie gli spazi a destra	**MyVar = RTrim(" vbscript ")** Il valore di MyVar è " vbscript".
Space	Crea una Stringa di spazi	**MyString = Space(10)** Restituisce una stringa con 10 spazi
Split	Crea un array di stringhe più piccole da una stringa	**MyString = "VBScriptXèXdivertente!"** **MyArray = Split(MyString, "x")** ' Il valore di MyArray(0) è "VBScript". ' Il valore di MyArray(1) è "è". ' Il valore di)MyArray(2) è "divertente!".
StrComp	Confronta il valore di due stringhe	**MyStr1 = "Cane"** **MyStr2 = "Gatto"** **MyStr3 = "Gatto"** **MyComp = StrComp(MyStr3, MyStr2)** Restituisce 0. **MyComp = StrComp(MyStr1, MyStr2)**

		Restituisce -1. **MyComp = StrComp(MyStr2, MyStr1)** Restituisce 1.
String	Crea una stringa di un carattere ripetuto un certo numero di volte	**MyString = String(5, "*")** Restituisce "*****". **MyString = String(5, 42)** Restituisce "*****". **MyString = String(10, "ABC")** Restituisce "AAAAAAAAAA".
StrReverse	Inverte una stringa	**MyStr = StrReverse("VBScript")** Il valore di MyStr è "tpircSBV".
Trim	Taglia gli spazi da entrambi i lati di una stringa	**MyVar = Trim(" vbscript ")** Il valore di MyVar è "vbscript".
Ucase	Converte in maiuscolo	**MyWord = UCase("Salve gente!")** Restituisce "SALVE GENTE!".

Nell'esempio seguente vengono viste alcune delle funzioni qui sopra descritte

```
<script language="VBScript">
<!–
Option Explicit
dim nome,i
nome=inputbox("Inserisci il tuo nome")
document.write "Il tuo nome ha " &len(nome) & " lettere<br>"
document.write "Le prime tre lettere sono: " &left(nome,3) & "<br>"
document.write "Le ultime tre lettere sono: " & right(nome,3) & "<br>"
document.write "Il tuo nome al contrario è : " &strreverse(nome) & "<br>"
document.write "In maiuscolo: " &Ucase(nome) & "<br>"
document.write "In minuscolo: " &Lcase(nome) & "<br>"
document.write "Iniziale maiuscola: " &Ucase(left(nome,1)) &Lcase(mid(nome,2)) & "<br>"
for i =1 to len (nome)
  document.writemid(nome,1,i) & "<br>"
next
//–>
</script>
```

FUNZIONI DI CONVERSIONE

Ecco una carrellata di funzioni di conversione:

Abs	Restituisce il valore assoluto di un numero	**MyNumber = Abs(50.3)** Restituisce 50,3 **MyNumber = Abs(-50.3)** Restituisce 50,3
CByte	Converte in un sottotipo Byte	**MyDouble = 125.5678** MyDouble è un valore di tipo Double. **MyByte = CByte(MyDouble)** Il valore di MyByte è 126.

CDate	Converte in un sottotipo Date	**MyDate = "October 19, 1962"** Definisce la data. **MyShortDate = CDate(MyDate)** Converte nel tipo di dati Date. **MyTime = "4:35:47 PM"** Definisce l'ora. **MyShortTime = CDate(MyTime)** Converte nel tipo di dati Date.
CDbl	Converte in un sottotipo Double	**MyCurr = CCur(234.456784)** MyCurr è un valore di tipo Currency (234.4567). **MyDouble = CDbl(MyCurr * 8.2 * 0.01)** Converte il risultato in un valore di tipo Double (19.2254576).
CInt	Converte in un sottotipo Integer	**MyDouble = 2345.5678** MyDouble è un valore di tipo Double. **MyInt = CInt(MyDouble)** Il valore di MyInt è 2346.
CLng	Converte in un sottotipo Long	**MyVal1 = 25427.45: MyVal2 = 25427.55** Sono valori di tipo Double. **MyLong1 = CLng(MyVal1)** Il valore di MyLong1 è 25427. **MyLong2 = CLng(MyVal2)** Il valore di MyLong2 è 25428.
CSng	Converte in un sottotipo Single	**MyDouble1 = 75.3421115: MyDouble2 =75.3421555** **MySingle1 = CSng(MyDouble1)** Il valore di MySingle1 è 75,34211. **MySingle2 = CSng(MyDouble2)** Il valore di MySingle2 è 75,34216.
CStr	Converte in un sottotipo String	**MyDouble = 437.324** MyDouble è un valore di tipo Double. **MyString = CStr(MyDouble)** Il valore di MyString è "437,324".
Fix	Restituisce la parte intera di un numero	**MyNumber = Fix(99.2)** Restituisce 99. **MyNumber = Fix(-99.8)** Restituisce -99. **MyNumber = Fix(-99.2)** Restituisce -99.
Hex	Restituisce la stringa che rappresenta il valore esadecimale di un numero	**MyHex = Hex(5)** Restituisce 5. **MyHex = Hex(10)** Restituisce A. **MyHex = Hex(459)** Restituisce 1CB.

Int	Restituisce la parte intera di un numero	MyNumber = Int(99.8) ‘ Restituisce 99 MyNumber = Int(-99.8) ‘ Restituisce -100 MyNumber = Int(-99.2) ‘ Restituisce -100.
Oct	Restituisce una stringa che rappresenta il valore ottale di un numero	MyOct = Oct(4) ‘ Restituisce 4. MyOct = Oct(8) ‘ Restituisce 10. MyOct = Oct(459) ‘ Restituisce 713
Round	Arrotonda ad un numero di decimali specificato	DimMyVar, pi pi = 3.1415 MyVar = Round(pi, 2) ‘ Il valore di MyVar è 3,14
Sgn	Restituisce un intero che rappresenta il segno di un numero	MyVar1 = 12: MyVar2 = -2.4: MyVar3 = 0 MySign = Sgn(MyVar1) ‘ Restituisce 1. MySign = Sgn(MyVar2) ‘ Restituisce -1. MySign = Sgn(MyVar3) ‘ Restituisce 0.