Ken’s Visual Basic Notes

Contents

Variables

Dim | Private | Static varname [**As** type]

Typical variable declarations

```
Dim sErrorMessage As String
Dim sUserName As String * 25
Dim iIndex As Integer, fInterestRate As Single
Private bAddMode As Boolean
Public iUserStatus As Integer
Static iRunningValue As Integer
```

Keyword	Scope
Dim	Variable only available within the procedure or module that declares it. When decleared within the Declarations section at the start of a module, it's known as a module-level variable.
Private	Can only be coded in the declarations section of a module. Variable is only available to the procedures within the module that declares it. (m)
Public	Can only be coded in the declarations section of a module. Variable is available to the procures in all the modules of a project. (g)
Static	Variable only available within the procedure or module that declares it. Variable retains its value from one execution of the procedure to another

```
dStartDate = #January 1, 2008#   ' use # delimiters for dates
```

Constants

[**Private | Public**] As **Const** constname [**As** type] = expression

Typical declarations for user defined contants

```
Private Const iMonths As Integer = 60
Public Const gfInterestrate As Single = .125 / 2
```

DataTypes

Datatype	Prefix		Suffix Literal
Boolean	bln or b	True (+1 or –1) or False (0)	
Byte	byt or t	0 to 255	
Currency	cur or c		86.00@
Date	dtm or d	*must embed literal date & time values inside # signs*	#12-Jan-1999# #2:56 PM#
Double	dbl or p	float (double precision)	86.0#
Integer	int or i	-32,768 to 32,767	
Long	lng or l	long integer	86&
Object	obj or o		
Single	sng or f	float	86.0!
String	str or s	Dim aString As String * 20	
Variant	vnt or v		

IsEmpty(anArg)	returns True if has not been initialized since declaring
IsNull(anArg)	returns True for (e.g.) empty string
IsNumeric(aString)	return True if can be converted to numeric type
VarType(anArg) **VarType**(aString)	returns what this argument can be converted into 0=vbEmpty 1=vbNull 2=vbInteger 3=vbLong 4=vbSingle 5=vbDouble 6=vbCurrency 7=vbDate 8=vbString 9=vbObject 10=vbError 11=vbBoolean 12=vbVariant 13=vbDataObject (DAO database access) 14=vbDecimal 17=vbByte 8192+int=vbArray array argument of the type specified by the int addition to 8192

int = **Asc**(str)	ascii number of first char in string
int = **CInt**(4.92)	rounds to next highest int
str = **CStr**(4.92)	
int = **Fix**(4.92) 'returns 4	truncate fraction
int = **Int**(4.92) 'returns 4	rounds down
Hex(11) 'returns B	
Oct(8) ' returns 10	

Str(10) ' returns string 10	
Val("10") ' returns 10	string to integer

Binary data conversion functions: AscB, InStrB, MidB, LeftB, RightB, ChrB

Arrays

```
Dim iValues (9) As Integer          ' index values are 0-9
Dim iValues (1 To 10) As Integer  ' index values are 1-10
Dim iValues (4, 4) As Integer        ' two dimensional array
Dim iValues (1 To 5, 1 To 5) As Integer   ' another two dimensional array

Dim cRate (1 To 10) As Single
cRate(1) = 0.01

Dim cRate (1,4) As Single

cRate(0,0) = 0.01
```

Strings

& (or +) concatenate

Str(10) ' returns string 10	integer to string
Val("10") ' returns 10	string to integer

Format

aVariant = Format(expression, strFormat)

strFormat	
"Currency"	Format(1234.678, "Currency")
"Fixed"	displays at least one digit before and two digits following the decimal point, with no thousandths separator
"General Number"	displays the number with no thousandths separator
Medium Time"	displays the time in 12 hour format
"On/Off"	display "On" if the value contains a nonzero value
"Percent"	displays the number, multiplied by 100, and adds the percent sign to the right of the number
"Scientific"	
"Short Time"	display the time in 24 hour format
"True/False"	
"Yes/No"	

Format Codes

txtAmount = **Format** (curAmount * interestRate, "$###,##0,00")

' if the value happens to be $0.00, the zeros ensure that the value prints, whereas if you used a # in place of each 0, the # would show nothing if the result were zero

txtAmount = **Format** (curAmount * interestRate, "$#####.000") 'displays to 3 places
txtAmount = **Format** (curAmount * interestRate, "$#####") 'displays no decimal places

aChar = **Chr**(int)	
aSubString = **InStr**([start,] bigstr1, littlestr2)	returns porition of littlestr2 within bigstr1, beginning at start if specified
aSubString = **InStrRev**(bigstr1, littlestr2 [, start,])	
aString = **Join** (aStringList [,delimiter])	default delimiter is a space
aString = **LCase**(str) aString = **UCase**(str)	case
size = **Len**(str)	length (works on numbers too)
aSubString = **Mid**(str, intStart [intLen]) '12345 strSentence = "I flew home"	careful – there's both a Mid function and Mid statement Mid statement replaces part of a string with

Mid(strSentence, 3, 4) = "rode" 'I rode home	another value
aString = **Month**(month [,abbre]) aString = **WeekdayName** (weekday [,abbre])	abbre = True for abbreviated month
aString = **Right** (str, int) aString = **Left**(str, int)	returns int far right characters
aString = **Str** (int) anInt = **Val**(str)	
aString = **Trim**(str) aString = **RTrim**(str) aString = **LTrim**(str)	removes blanks at the ends
InStr	Returns the occurrence of one string within another

Date Time

isDate(aDate) ' is it a valid date?

Date intYear = Year(Date) ' returns current year	returns current date
DateSerial(intYear, intMonth, intDay) dte = **DateSerial**(1998, 7, 18) 'July 18, 1998 dte = **DateSerial**(1998, 7, 18-31) 'returns 31 days prior to July 18, 1998	returns the date corresponding to ints passed in
DateAdd(strIntrv1, intN, dteDate) **DateAdd**("d", 20, dteUserDate) 'returns 20 days beyond dteUserDate strIntrv: "h"=hour, d=day, m=month, n=minutes, q=quarter, s=second, y=day of year, w=weekday, ww=week, yyyy=year	adds the intN value to the date specified by dteDate for the given strIntrv1
DateDiff(strIntrv1, dte1, dte2) no_of_weeks = DateDiff("ww", dte1, dte2)	returns the number of time intervals (specified by strIntrv) between the two dates
Now	returns the current date-time in the date format
Time	returns the current time
Timer lngBefore = Timer … lngAfter = Timer lngTimeDiff = lngAfter – lngBefore 'diff in seconds	returns the number of seconds since midnight
TimeSerial(hour, min, sec) dteTimePaid = **TimeSerial**(14, 32, 25) '2:32:35 P.M. aDte = **TimeSerial**(14-20, 32, 25)	return the date and time in the internal date format for the time specified

'20 hours before 2:32:35 P.M.	

Operations

aRoot = aFloat ^ (1/5) ' raise to a power
Order: mult,div,add,sub first, then left to right evaluation

Logical Expressions

if theAnswer = "yes"

Relational Operators: =, <>, >=, >, <=, <

Logical operations: Not, And, Or, Xor, Eqv, Imp

> The two very rarely used VB 6 operators Imp and Eqv have disappeared in VB.NET, but two new operators, AndAlso and OrElse have been added

Decisions

```
If   (aNumber = anotherNumber)  Then     ' not    = =
If   aNumber < > anotherNumber           Then    ' not    ! =
If   Not (aNumber < > anotherNumber )  Then    ' not    ~
```

"Apple" <= "Orange" ' true

```
If    some_comparison   Then

Else

End If
```

```
If (A > B) And (C < D) Then
If (A > B) Or (C < D) Then
If Not (StrAns = "Yes") Then
```

```
Select Case iIndex
        Case 0,2
         …
        Case 1
        …
        Case 10 To 24
        …
        Case Is >= 25
        …
        Case Else
        …
End Select
```

Looping

```
Do While (  )
…
Loop
```

```
Do Until ( )
…
Loop
```

```
Do
…
Loop Until ( )
```

```
For i = startVal   To endVal  Step 1    ' Step 1 is optional
…
Next i
```

Collections, Iteration

```
Public aList As New Collection

aList.Add "Ken"
aList.Add "Sally"

int = aList.Count

aList.Remove 3  ' removes the third item

For Each aControl In Controls
  aControl.Visible = True
Next aControl
```

Subroutines, Functions

```
Public Sub aSub(aString As String,   ' default is to pass as a ref
                x as Int,
                ByVal Y as Int )              ' pass by value
        Dim maxX As Int
        Dim maxY As Int

        x = x + 1   ' changes x in the callers space
        y = y + 1  ' does NOT change in the callers space
        …
End Sub

aSub(string1, anInteger)        ' calling a private subroutine w parms passed by reference
Module3. aSub(string1, anInteger)   ' calling w fully qualified name

Sub foo (parm1 As String, Optional parm2 As Integer = 1)  ' optional parm
…
End Sub

Private Sub cmdCompute_Click ()             ' subroutines do not return a value
…
End Sub


Public Function ErrorCheck() As Integer     ' functions return a value
…
  ErrorCheck = 1    ' set return value
  Exit Function
…
End Function
```

Error Handling

Syntax	Meaning
On Error GoTo *label*	On error goto the line following the label
On Error Resume Next	Resume processing on the next statement after the one that caused the error
On Error Goto ( )	Disable the error handling code
Resume	Use this in an error handling routine to resume processing with the statement that caused the error
Resume Next	Use this in an error handling routine to resume processing after the statement that caused the error
Exit Sub	Return to the caller

```
Private Sub somesub_Click ( )
        On Error GoTo ErrorHandler
        …
        Exit Sub

        ErrorHandler:
                MsgBox "Error Number: " & Err.Number & vbCrLf _
                 & " Error Description: " & Err.Description
End Sub
```

Summary of Other Statements

Statements for working with forms

Load	Load a form into memory
Unload	Unloads a form

Statements for working with string variables

Lset	Left align a string within a string variable
Mid	Replaces a specified number of characters in a string variable with characters from another string
Rset	Right aligns a string within a string variable

Statements for controlling program execution

End	Terminates a program immediately and releases all resources
Stop	Sets a breakpoint in the code that suspends execution
Exit	Provides a way to prematurely exit from a Do loop, For loop, Sub procedure or function when used within these forms: Exit Do, Exit For, Exit Sub, or Exit Function

Miscellaneous statements

Type	Often used to define the fields within the records of a structure

Summary of Visual Basic Functions

Some of the date and time functions

Date	Returns the current date
Now	Returns the current date and time
Time	Returns the current time
DateValue	Receives a date string and returns a date numeric value

Some of the math functions

Int	Returns the integer portion of a number
Val	Returns the number contained in the string as an numeric value
Rnd	Returns the next random number as a decimal franction between 0 and 1

The immediate If function

IIf	Evaluates a conditional expression and returns ome value if the epxression is True and another value if the expression is False

Two dialog functions

MsgBox	Displays a message in a dialog box, waits for the user to click a button and returns an integer indicating which button the user clicked If MsgBox("Want to Exit?", _ vbYesNo + vbDefaultButton, "Confirm Exit") = vbNo Then …
InputBox	Returns a string that contains what the user entered msgText = InputBox("Enter some words"_ , "InputBoxDemo"); if msgText = "" Then Exit Sub

File I/O

```
Open "d:\data\myfile.dat" For Append As #1
Open "d:\data\myfile.dat" For Input As #2
Open "d:\data\myfile.dat" For Output As #3

int_next_available_file_no = FreeFile()

Write #1, "some line to write"
Write #1, "end w semicolon to not insert newlinee";

Do Until (Eof(#1) = True)
   Input #1, parm1, parm2, parm3     'read comma separated parms
Loop

Do Until (Eof(#1) = True)
   Line Input #1, aStringLine   ' read a whole line
Loop

Type Record  ' user defined type
  ID As Integer
  Name As String * 20
End Type

Dim aRecord As Record  'declare variable

Do Until (Eof(#1) = True)
   Input #1, aRecord      ' reads the record
Loop
```