Miscellaneous Topics In Macro Programming

Agenda

- Speaker Introductions
- Library Management
- Copy by reference versus copy by value
- Advanced data structures
- Common error with Select Case
- Using the correct text object
- Q & A
- Examples tested with a 2.0 development build

Who Am I?

- I am Andrew Pitonyak
  - *Programmer / Architect – Java, C++, Perl, StarBasic*
  - *Using Open Office since StarOffice 5.x*
  - *Author of “OpenOffice.org Macros Explained” and “Andrew's Macro Document”*
  - *Four university degrees (Math and Computers)*
  - *Puppeteer*
  - *Firearms instructor*
  - *General class amateur radio license*
  - *The Technical University of Dresden*
  - *Husband to a wonderful woman (just ask my wife)*

How And Why Am I Involved?

- I wanted macros for myself.
  - *Little to no macro documentation existed*
  - *No one seemed to know how to use macros*
  - *Early 2003: started collecting examples for myself*
  - *April 29, 2003: Released “Andrew's Macro Document”*
  - *July 2, 2003: Whil Hentzen requested authors*
  - *July 2004: “OpenOffice.org Macros Explained”*

Library Management

- *Document and Application Libraries*
- The Standard library is special
  - *Automatically created*
  - *Can not delete the library, only the modules*
  - *Can not append to a new location*
  - *Always loaded*
- Use meaningful names
  - *Avoid confusion*
  - *Difficulty renaming*
  - *The libraries are stored as XML as explained in "OpenOffice.org XML Essentials—Using OpenOffice.org's XML Data Format" (see http://books.evc-cit.info/)*

Copy By Reference/Value

- Copy by value
  - Each variable contains its own copy
  - Changes in one do not affect the other

- Copy by reference
  - Two variables reference the same data
  - Changes in one are seen in the other

- Intrinsic types copy by value
  - *Integer, floating point, date, string*

Arrays Copy By Reference

- Array variables copy by reference
  - *I can assign one array to another and they reference each other (see page 37 of my book).*
  - *No problem with arrays of the same type if you remember the behavior.*

- Arrays of different type and dimension cause problems! *Consider the arrays a(), d(), and e().*

```
Dim a(1 To 2) As Integer
Dim d(-3 To -2, 1) As String
Dim e(-3 To -2) As String

d(-3, 0) = "one" : d(-2, 0) = "two"
d(-3, 1) = "eins" : d(-2, 1) = "zwei"
```

Use The Arrays

```
                   REM Assign two dimension array d()
a() = d()          REM to one dimension array a()
Print LBound(a(), 1) & " to " & UBound(a(), 1)  REM -3 to -2
Print LBound(a(), 2) & " to " & UBound(a(), 2)  REM  0 to  1

REM Although the assignment works, things
REM are not consistent
Print d(-2, 0)   REM "two"
Print a(-2, 0)   REM Compile time error
Print a(1)       REM Runtime error, index out of defined range

REM I can assign a string array to an integer array
REM ReDim Preserve does not preserve because of types
a() = e()        REM Assign to a one dimensional array
Print a(-3)      REM "one"
```

Structures Copy By Reference

- This is exactly what you want.
- Set the first paragraph's language to French by modifying the Locale structure.

```
Dim oCursor, aLocale
oCursor = ThisComponent.getText().createTextCursor()
oCursor.gotoStart(False)
oCursor.gotoEndOfParagraph(True)
oCursor.CharLocale.Language = "fr"
```

- *Why does this fail?*

Struct From Service Copies By Value

- From a Universal Network Object Service, A structure is returned as a copy.
- CharLocale is a com.sun.star.lang.Locale structure returned from a UNO service.
- Get a copy of the structure.
- Modify the copy.
- Copy back the modified structure.

```
oCursor = ThisComponent.getText().createTextCursor()
oCursor.gotoStart(False)
oCursor.gotoEndOfParagraph(True)
aLocale = oCursor.CharLocale          REM Make a copy
aLocale.Language = "fr"               REM Modify the copy
oCursor.CharLocale = aLocale          REM Copy back
```

UNO Service Copies By Reference

- Universal Network Object Service copies by reference

- From a Universal Network Object:
  - *A structure is returned as a copy.*
  - *A service is almost always returned as a reference.*

- I know two properties that return as a value

```
oDoc.StyleFamilies.PageStyles.Standard.TextColumns
oDoc.TextSections(0).UserDefinedAttributes
```

- I do not know why or how to tell.

Arguments Copy By Reference

- No matter what the type is.
- This is usually what you want.
- The behavior is different depending on how the argument is declared. (I will mention this again shortly)
- If you forget, you may have unusual unexpected bugs in your code...

Consider The Routine

```
Sub incrementNumber(n As Integer)
    Do While n < 5
        Print n
        n = n + 1
    Loop
End Sub
```

- Now, use the routine!

```
Dim i As Integer, d As Double
i = 0 : d = 0
incrementNumber(i)      REM Changes the value of i

incrementNumber(0)      REM No return, prints 0, 1, 1, ...

incrementNumber(d)      REM No return, prints 0, 0, 0, ...
```

Use The ByVal Keyword

```
Sub incrementNumber(ByVal n As Integer)
    Do While n < 5
        Print n
        n = n + 1
    Loop
End Sub
```

- All three calls now return.
- Changed the behavior – argument is unchanged.
- For variables copied by reference (array, struct, etc.), ByVal prevents the reference from changing, even though you can change the data.

Use A Temporary Variable

```
Sub incrementNumber(n As Integer)
    Dim nTemp As Integer
    nTemp = n
    Do While nTemp < 5
        Print nTemp
        nTemp = nTemp + 1
    Loop
    n = nTemp
End Sub
```

- All versions now work.
- Value is passed back for integer argument.

- If n is a variant, then the first version works with no changes!

Advanced Data Structures

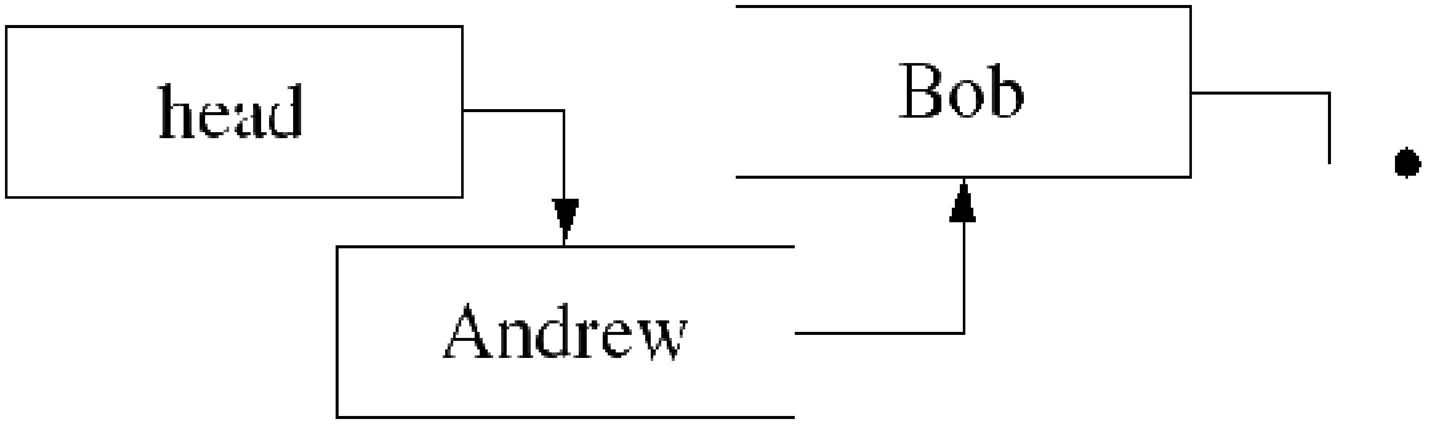
- I was told that you can not create advanced data structures in StarBasic.
- We can define our own structures.
- A Variant can point to anything.
- Although a user defined struct can not contain an array, a variant can reference an array.
- Looks like a node in a linked list...

```
Type PersonNode
    FirstName As String
    NextNode  As Variant
End Type
```

Create A Linked List

- The head node contains no data!

```
Dim headNode As PersonNode
Dim Person1 As PersonNode
Dim Person2 As PersonNode
Person1.FirstName = "Andrew"
Person2.FirstName = "Bob"
headNode.NextNode = Person1
Person1.NextNode = Person2
```



Search A Sorted List

```
Function findNode(ByVal head As PersonNode, sName$)
    Do While NOT IsEmpty(head.NextNode)
        If head.NextNode.FirstName >= sName Then
            Exit Do
        End If
        head = head.NextNode
    Loop
    findNode = head
End Function
```

- Returns the node before the node that should contain the value (even if it does not exist).
- The head is passed by value! (ByVal keyword)
- I want short circuit evaluation.

Add A Node In A Sorted List

- Add a node using CreateObject()

```
Function addNode(ByVal head As PersonNode, sName$)
    Dim c, tempNode
    c = findNode(head, sName)
    If NOT IsEmpty(c.NextNode) Then
        If c.NextNode.FirstName = sName Then
            addNode = c
            Exit Function
        End If
    End If
    tempNode = CreateObject("PersonNode")
    tempNode.FirstName = sName
    tempNode.NextNode = c.NextNode
    c.NextNode = tempNode
    addNode = c
End Function
```

Create A Sorted Linked List

```
Dim headNode As PersonNode
addNode(headNode, "Bob")
addNode(headNode, "Andrew")
addNode(headNode, "Michelle")
```

- Structs copy by reference.
- Copy by value is a problem for UNO properties.
- Accessing NextNode from the structure does not copy by value.

```
REM This works as expected
headNode.NextNode.FirstName = "Bobby"
```

Select Case Is Easy

```
Dim i As Integer
i = Int(25 * Rnd() - 10)
Select Case i
    Case IS = 1, IS = 3, IS = 5
        Print "i is [1, 3, 5] = " & i
    Case 6 To 10
        Print "i is [6 to 10] = " & i
    Case < -5
        Print "i is less than -5 = " & i
    Case IS > 10
        Print "i is greater than 10 = " & i
    Case Else
        Print "Sorry, i = " & i
End Select
```

Select Is Frequently Wrong

- Understand the simple mistake.

```
Select Case i   REM i is an integer
Case i=-1       REM Works for i=-1, i=0

Case i<>0       REM Works for i=-1, i=0

Case i=2        REM Works for i=-1, i=0 (fails for i=2)
```

- HINT, this works!

```
i = 2
Select Case True
Case i=2    REM Works ONLY for i=2
```

More Common Mistake

- A more common mistake

```
Select Case  i
    Case  i > 2 AND i < 10
```

- Excellent solutions available else where
  - *My macro document, see “**Select Case**”*
  - *My book, see pages 58 and following*
  - *Bernard Marcelly has an excellent solution*

Get A Text Object

- A Write document contains a text object
  ThisComponent.getText()

- Other objects also contain a text object
  - *XTextRange – cursors, anchors, text table cell.*
  - *XTextContent contains an anchor – text tables, text sections, text fields, text frame, text section, graphics objects*

Use The Correct Text Object

- A cursor traverses the text object that created it.
- *A text object can create a cursor for a range that it contains.*
- *A text object can compare cursors it contains.*

```
Dim oCursor, oTable, oCell, oText
oTable = ThisComponent.getTextTables().getByIndex(0)
oAnchor = oTable.getAnchor()
oText = ThisComponent.getText()
oCell = oTable.getCellByPosition(0, 0)
REM oText.createTextCursorByRange(oCell)
oCell.getText().createTextCursorByRange(oCell)
```

Discount From The Publisher

- 40% conference discount price on ALL books purchased directly from Hentzenwerke.
- http://www.hentzenwerke.com
- Step 4 of the checkout process, there is a box for a “Promotion Code”.
- Promotion Code is “OOOBERLIN”
- Questions?