ASSOC CUSTOM CONTROL V0.01 Copyright © 1994 Axiomatic Software Limited
Written by Andrew Cheshire, CIS 100273,1543

NB: This control is not compatible with VB 1.0 -- you won't be able to add it to your project.

ASSOC is a Visual Basic custom control. Like the Timer control (but unlike most custom controls) it does not display anything on the screen at runtime; it was written to implement a useful data-structure which is not implemented in Visual Basic.

The word **ASSOC** stands for "associative array". This is a data-structure which is sometimes known as a "dictionary" -- it's like an array where the subscripts are strings instead of numbers.

For example, consider the following program which reads in names and addresses from a text file (with the name and the address on alternating lines to simplify the example) ..

```
Dim Lastnames(1 To 100) As String, Addresses(1 To 100) As String

Sub TryArray
Dim Lastname As String, Address As String
Dim I As Integer, N As Integer
Open "ADDR.DAT" For Input As File #1
For I=1 To 100
        If EOF(1) Then N=I-1 : Exit For
        Input #1,Lastname
        Input #1,Address
        Lastnames(I)=Lastname
        Addresses(I)=Address
Next I
Close #1
End Sub
```

Lastnames would then be an array of strings representing people's names and *Addresses* would be an array of strings representing their addresses. But if you wanted the address of "Smith" you'd have to find it like this ..

```
For I=1 To N
        If Lastnames(I)="Smith" Then
                theAddress=Addresses(I)
                Exit For
        End If
Next I
```

Suppose Visual Basic let you use strings as subscripts -- you could initialise a single array like this ..

```
Dim Addresses(String) As String
Dim Lastname As String, Address As String
Open "ADDR.DAT" For Input As File #1
Do Until EOF(1)
        Input #1,Lastname
        Input #1,Address
        Addresses(Lastname)=Address
Loop
Close #1
```

.. and you could find the address of "Smith" like this ..

```
theAddress=Addresses("Smith")
```

The **ASSOC** control provides this facility but with a different syntax. If you had an **ASSOC** called *Addrs* on your form the above example would look like this ..

```
Sub TryAssoc
Dim Lastname As String, Address As String
Open "ADDR.DAT" For Input As File #1
Do Until EOF(1)
        Input #1,Lastname
        Input #1,Address
        Addrs.Key=Lastname
        Addrs.Value=Address
Loop
Close #1
End Sub
```

.. and to find the address of "Smith" ..

```
Addrs.Key="Smith"
theAddress=Addrs.Value
```

The TryArray and TryAssoc routines are in the *EXAMPLE.MAK* project.

MORE DETAIL

So .. this control can be used for storing and retrieving key-value pairs. To store an item you set the *Key* property and then set the *Value* property. To retrieve an item you set the *Key* property and then read the *Value* property. If you want to use numbers in it instead of strings just go ahead and Visual Basic will sort it out (well, version 3 will anyway, not sure about earlier versions) ..

```
Assoc1.Key=34
Assoc1.Value=97
```

There is only ever one entry for a given key -- for example, after ..

```
Addrs.Key="Smith"
Addrs.Value="19 Windsor Road"
...
Addrs.Key="Smith"
Addrs.Value="23 Buckingham Street"
```

.. there is only one entry keyed by "Smith" and that is "23 Buckingham Street", the first entry has been overwritten.

If you try to retrieve a value for a key which has not been set you get an empty string ..

```
Addrs.Key="Humperdink"
Address=Addrs.Value
```

.. if no entry for "Humperdink" has been previously made the *Address* variable gets set to "".

Trying to set or get a value when the key is set to an empty string as a key is not allowed (it raises

a Visual Basic program error).

Using an empty string as a value is ok but you can't immediately tell whether there is a value of "" or whether there is no value at all. A property called *Defined* returns TRUE if there is a value associated with the current key and FALSE otherwise. It is illegal to try to read the *Defined* property when *Key* is set to "".

You can set an entry using a single statement by means of the *KeyValue* property. For example ..

```
Addrs.KeyValue="Smith=19 Windsor Road"
```

You can also retrieve using this property ..

```
Addrs.Key="Smith"
Address=Addrs.KeyValue
```

.. sets *Address* to "Smith=19 Windsor Road".

Since *KeyValue* is the default property of the **ASSOC** control you can omit it's name ..

```
Addrs="Smith=19 Windsor Road"
```

The "=" in those strings is just a default -- you can change it using the *Delimiter* property ..

```
Addrs.Delimiter=" of "
Addrs.KeyValue="Smith of 9 Windsor Road"
```

If you set *Delimiter* to "" it gets set to " " (a single space). The *Delimiter* property appears in the Properties window and can be set at design time.

Note that a quick way to make an entry with a value of "" is ..

```
Assoc1.KeyValue=key
```

THE ACTION PROPERTY

The *Action* property is a property which cannot be read but which can be set to an integer to trigger some action.

Setting *Action* to 0 starts an "enumeration" -- see below.

Setting *Action* to 1 clears the control -- all key/value pairs are deleted and *Key* is reset to "".

Setting *Action* to 2 deletes the entry given by the current setting of *Key*. It is illegal to attempt to delete an entry when *Key* is set to "".

SCANNING THROUGH ITEMS

Items are actually sorted in ascending order by their keys as they are entered into the control. For example, after ..

```
Addrs.KeyValue="Jones=23 Cardiff Row"
Addrs.KeyValue="Butler=102 Stirling Lane"
Addrs.KeyValue="Smith=19 Windsor Road"
```

.. the entries are sorted in the following order ..

```
"Butler" "102 Stirling Lane"
"Jones" "23 Cardiff Row"
"Smith" "19 Windsor Road"
```

This ordering can be used in two ways. Firstly there is a property called *NextKey* which will return the key of the item following the current setting of *Key*. For example, after ..

```
Addrs.Key="Butler"
k=Addrs.NextKey
```

.. *k* is set to "Jones".

The *Key* property need not be set to the key of an existing item for this to work -- for example, after ..

```
Addrs.Key="E"
k=Addrs.NextKey
```

.. *k* is again set to "Jones"

To find the key of the very first item set *Key* to "" and then read *NextKey*. If there is **no** item whose key follows *Key* then *NextKey* is set to "". So to go through all the items you can do the following ..

```
Addrs.Key=""
Do
        Addrs.Key=Addrs.NextKey
        If Addrs.Key="" Then Exit Do
        Debug.Print Addrs.Key,Addrs.Value
Loop
```

Actually there is a better way to scan through **all** the items. If you set *Action* to 0 a custom event procedure called *Enumerate* is called for each item. This procedure is passed the *Key* and *Value* as parameters ..

```
Sub Assoc1_Enumerate (Key As String, Value As String)
        Debug.Print Key,Value
End Sub
```

SOME SUGGESTED APPLICATIONS

Find duplicate files on a disc ..

```
For each directory ..
        For each file ..
                Assoc1.Key=Filename
                If Assoc1.Defined Then
                        Assoc2.KeyValue=Filename
                Else
                        Assoc1.KeyValue=Filename
                End If
```

.. *Assoc1* is used to keep a list of *all* filenames. *Assoc2* is used to keep a list of filenames which appear more than once. Note the trick of assigning to the *KeyValue* property without bothering with a delimiter or a value since we only want a value of "" anyway. And remember that *KeyValue* is the default property and could have been omitted but I left it in to make the example clearer.

Read records from a file and sort them into order using the 10'th through 19'th characters as a sort key (assumed unique in this example) ..

```
Dim record As String, NL As String, S As String

Open "Test" For Input As #1

Assoc1.Action = 1 ' clear all keys and values

Do Until EOF(1)
        Input #1, record
        Assoc1.Key = Mid(record, 10, 10)
        If Assoc1.Defined Then Stop
        ' - a duplicate key
        Assoc1.Value = record
Loop

Close #1

' The following code writes the sorted records to a text control
' (the text control must be multiline)

NL = Chr(13) & Chr(10)

S = ""
Assoc1.Key = ""
Do
        Assoc1.Key = Assoc1.NextKey
        If Assoc1.Key = "" Then Exit Do
        S=S & Assoc1.Value & NL
Loop
Text1.Text = S
```

SO WHAT DO YOU THINK?

I have written this custom control as an experiment. Visual Basic has a hell of a lot going for it but it's a bit short on interesting data-structures and doesn't provide you with the ability to devise your own. **ASSOC** begins to fill that gap. Ok, you could use the new database engine instead but it seems over the top for the sort of applications that **ASSOC** is targeted at.

If you're find **ASSOC** useful please let me know, either by mail or in the *Programming Issues* section of the *MS BASIC* forum. If the response is positive I can enhance **ASSOC** and write more controls of this sort.

By the way, **ASSOC** stores the dictionary items in a very simple-minded fashion at the moment (foolish to attempt optimisation when there might be bugs in it). So if it runs slowly on your PC as the number of items gets large don't worry -- if enough people are interested for me to write another version it'll be a **lot** faster for large numbers of items.

I suppose it's only fair to mention that future controls I write might be shareware rather than freeware.

LEGAL STUFF

This software is provided 'AS IS' and the author makes NO WARRANTY or representation, either express or implied, with respect to the software, its quality, accuracy or fitness for a particular purpose.

This software is copyright © 1994 Axiomatic Software Limited.
All rights reserved, except as specified below.

Permission is hereby granted to use, copy and distribute this software for any purpose, without fee, subject to these conditions:

(1) The software may only be distributed in its entirety; in particular this notice must be included unaltered. No fee may be charged.

(2) Permission for use of this software is granted only if the user accepts full responsibility for any undesirable consequences; the author accepts NO LIABILITY for damages of any kind.

SUMMARY OF PROPERTIES, EVENTS AND METHODS

PROPERTIES

Action	set to 0 for Enumeration, 1 to clear all, 2 to delete current item
Defined	gives TRUE if there is a value associated with the current key
Delimiter	the delimiter string separating key from value in the *KeyValue* property
Key	the key used to identify the item
KeyValue	both key and value, delimited by a string you can specify
NextKey	the key of the item following the current key
Value	the data associated with the current item

EVENTS

Enumerate	is called once for each item when *Action* is set to 0

METHODS

(none)