Successive Approximation (Binary Chop) Search Algorithm

This module is part of the original MMBasic library. It is reproduced here with kind permission of Hugh Buckle and Geoff Graham. Be aware it may reference functionality which has changed or is deprecated in the latest versions of MMBasic.

BSEARCH.BAS

```
' Binary Search Routine  - From Hugh Buckle, Jan 2012
' Searches for an entry in a sorted text array.

' You can search all or part of the array by setting
' StartIndx and EndIndx to appropriate values.  These will
' normally be the array extents, as they are here.

' If the subroutine BinSearch returns Found = 1 then
' the value can be found at a$(Indx).

' You could use this routine in conjunction with the Bubble Sort
' as I have done to ensure the array is in ascending order,
' but note the use of Lcase$() in both the bubble sort and
' binary search because I want the sequence to be not case
' sensitive, as in a dictionary and in text sorted in Word
' and Excel.

' To search a numeric array, just remove all the Dollar signs
' and provide some numeric Data.

' This is just some test data
Data Andrew,Fred,Geoffrey,George,Hugh,Isaac,aardvark
Data James,Joe,John,Lance,Victor,William,Aaron
Dim a$(20)
Print
j=14     ' number of names in the Data statements
For i = 1 To j
  Read a$(i)
Next

' Sort the list of names
GoSub BblSort
Print "Sorted Names:"
For i=1 To j
  Print i; " "; a$(i)
Next
Print
Print

' This Do loop invokes the search and prints the result
Do
```

```
        ' This example searches the whole array a$().
        ' To search part of the array, set StartIndx and EndIndx to
        ' the boundaries you want.
        StartIndx = 1
        EndIndx = j
        Input "Find (end to exit)"; b$
        GoSub BinSearch
        If found = 1 Then
          Print b$; " found at "; "a$("; indx; ")"
        Else
          Print b$; " wasn't found. Index = "; indx
        EndIf
      Loop While b$ <> "end"
      Print "Bye"
      End

BinSearch:
      ' The Binary search looks for b$ in array a$().
      ' It tests the value at the mid point of the array.
      ' If the value there is greater than the value we are looking for,
      ' the START point is set to where we just looked + 1.
      ' If it was less, then the END point is set to where we just
      ' looked - 1. Then the routine loops. So on each loop we halve
      ' the section of the array to search until the entry is found
      ' or either the Start or End point exceeds the array boundaries.

      ' Call this routine with
      '   Array a$() sorted in ascending sequence
      '   StartIndx and EndIndx set to the lower and upper limits
      '     of a$() you wish to test.
      '   b$ set to the value to find.

      ' BinSearch returns Found = 1 if the value was found, 0 if not.
      ' If found, it returns Indx pointing to the matching element in a$()

      Found = 0
      Do
        ' Find mid point of the section of the array to search
        Indx = StartIndx + Fix((EndIndx - StartIndx)/2)
        If LCase$(b$) = LCase$(a$(Indx)) Then
          Found = 1
          ElseIf LCase$(b$) > LCase$(a$(Indx)) Then
            StartIndx = Indx + 1
          Else
            EndIndx = Indx - 1
        EndIf
      Loop Until Found = 1 Or StartIndx > EndIndx Or EndIndx < StartIndx
      Return
```

```
BblSort:
    ' Bubble sort routine used to sort the list of names
      Flips = 1
      Do
        Flips = 0
        For n=1 To j-1
          If LCase$(A$(n)) > LCase$(A$(n+1)) Then
            SaveA$ = A$(n)
            A$(n) = A$(n+1)
            A$(n+1) = SaveA$
            Flips = 1
          EndIf
        Next
      Loop While Flips = 1
      Return
```

BSEARCH1.BAS

```
BinSearch:
      ' The Binary search looks for b$ in array a$().
      ' It tests the value at the mid point of the array.
      ' If the value there is greater than the value we are looking for,
      ' the START point is set to where we just looked + 1.
      ' If it was less, then the END point is set to where we just
      ' looked - 1. Then the routine loops. So on each loop we halve
      ' the section of the array to search until the entry is found
      ' or either the Start or End point exceeds the array boundaries.

      ' Note that this search routine expects the array A$() to be in
      ' sequence which is CASE INSENSITIVE. e.g. aa comes before Ab
      ' If the array was sorted case sensitive then remove Lcase$().
      ' Call this routine with
      '   Array a$() sorted in ascending sequence
      '   StartIndx and EndIndx set to the lower and upper limits
      '     of a$() you wish to test.
      '   b$ set to the value to find.

      ' BinSearch returns Found = 1 if the value was found, 0 if not.
      ' If found, it returns Indx pointing to the matching element in a$()

      Found = 0
      Do
        ' Find mid point of the section of the array to search
        Indx = StartIndx + Fix((EndIndx - StartIndx)/2)
        If Lcase$(b$) = Lcase$(a$(Indx)) Then
          Found = 1
          ElseIf Lcase$(b$) > Lcase$(a$(Indx)) Then
            StartIndx = Indx + 1
```

```
        Else
          EndIndx = Indx - 1
      EndIf
    Loop Until Found = 1 Or StartIndx > EndIndx Or EndIndx < StartIndx
    Return
```