The binary search algorithm is designed for fast searching of ordered arrays. It works by checking the middle element of the array, then deciding whether the item searched for is in the upper or lower half of the array. This reduces the section of the array that needs to be searched by at least one half. The process is repeated until the item is found or it is clear that the item is not in the array.

A binary search is generally much faster than an ordinary sequential search. It can be shown, that to locate an item in an array having N items, at most \( \log_2(N+1) \) comparisons will be needed, where “\( \log \)” is the base 2 log function. This means that in searching a list containing 1000 items, no more than \( \log(10001) \sim \log(1024) = \log(2^{10}) = 10 \) comparisons will need to be made!

```vbnet
'************************ Binary Search  ****************************
Action: Does a binary search of the sorted array List.
Parameters:
IN:      Item             The item to be searched for
IN:      List()           The array of DataType to be searched
IN:      First            The first subscript of List to consider
IN:      Last             The last subscript of List to consider
OUT:     ItemFound        Set to 1 if and only if Item was found in List
OUT:     Pos              If Item found, set to the array index such that
                          List(Pos) == Item.
                          If Item not found, Pos is set to the array index at
                          which Item should be.
Precondition: The array List MUST be in sorted natural order, with
              List(First) holding the smallest element and List(Last)
              holding the largest element.
Note: DataType must be a type that supports comparisons ( <, ==, > ).
---------------------------------------------------------------------------*/
Sub BinarySearch (ByVal Item As DataType, ByRef List() As DataType,
                  ByVal First As Integer, ByVal Last As Integer,
                  ByRef ItemFound As Integer, ByRef Pos As Integer)
    Dim Mid as Integer;  // Middle: used to “split” array
    ItemFound = 0;       //assume item not there
    if ( Item > List(Last) )  // Special case if Item is larger then
        Pos = Last + 1  // last location, then item not in the list.
        return;
    end if

    while ( Not ItemFound AND First <= Last )
        Mid = (First + Last)/2;  // Divide list in half
        if ( Item = List(Mid) )  // Item there, then done searching
            ItemFound = 1;
            Pos = Mid;
        else if ( Item > List(Mid) )  // Try the last half
            First = Mid + 1;
        else
            Last = Mid - 1;
        end while
    if ( NOT ItemFound )  // Item belongs in List[First]
        Pos = First;
    end Sub
```