LSearch Search cells for a match with specific data

#include <Lists.h> **List Manager Package**

Boolean	**LSearch(***dataPtr*, *dataLen*, *compProc*, *theCell*, *theList* **);**	
Ptr	*dataPtr* ;	address of data to match
short	*dataLen* ;	length of data to match
ProcPtr	*compProc* ;	address of comparison function (0=standard)
Cell	***theCell** ;	where to start; receives cell where found
ListHandle	*theList* ;	handle leading to a ListRec
	returns	was a match found?

LSearch examines the contents of cells in a list, attempting to find a cell containing specified data.

dataPtr is the address of some data to match. Unless you have written a custom 'LDEF', this is the address of some text.

dataLen is the length of the data to match.

compProc is the address of a pascal-style callback function that will be called repeatedly to compare the contents of each cell with the data at *dataPtr*. Use 0 to specify the default comparison function, which is **IUMagIDString**.

theCell is the address of a 4-byte Cell (a.k.a. Point). On entry, it specifies the first list element to compare. Upon return, it identifies the cell in which a match was found (if the returned value is TRUE**).**

theList is a handle leading to a variable-length ListRec structure. It is a value previously obtained via **LNew**.

Returns: a Boolean. It is one of:

FALSE	Match not found.
TRUE	A match was found.

Notes: The list is searched in row-major order, starting at *theCell* and advancing left-to-right and top-to-bottom as with **LNextCell(**TRUE,TRUE,...).

You can use **LSearch** as a lookup routine to locate a cell if you happen to know its contents. In this respect, the List Manager can be used in simple database operations. A more common use is to call **LSearch** followed by **LSetSelect** and **LAutoScroll** in order to pre-select a default (e.g., highlight the current font in a font-selection list).

```
SetPt( &theCell, 0,0 );                    /* search from top of list */
if ( LSearch( "Geneva", 6, NIL, &theCell, theList ) {
    LSetSelect( TRUE, theCell, theList );
    LAutoScroll( theList );
}
```

Callback Comparison Function

By default, **LSearch** uses **IUMagIDString** as the comparison function (a case-insensitive comparison that must match for the full length of the text). If you want to implement one-character "hotkey" searching (similar to the technique used by Standard File), you will need to write a custom callback routine. The format for such a comparison routine is:

```
pascal Boolean myCmpProc( char *cellData, char *testData, short cellLen,
    short testLen )
{
    /*. . . compare cellData and cellLen to testData and testLen. . .*/

    if ( /* . . . they match . . . */)   /* note inverted return logic */
        return( FALSE );
    else
        return( TRUE )
}
```

To implement a simple hotkey-style search routine, your comparison function should be less rigid than **IUMagIDString**. There are several non-trivial pitfalls, so a complete example is shown below. This example works best if the list elements are in alphabetical order.

Example

```
#include <Files.h>

/* Custom comparison routine matches input/output parms of IUMagIDString
    Matches if first character of cell is greater or equal to test string.
*/
pascal short  cmp1stChar( Byte *cellPtr, Byte *testPtr, short *cellLen,
        short testLen )
{
    if (cellLen==0 || testLen==0) return(1);
    if ( *cellPtr >= *testPtr ) return(0);  /* Return 0 for a match */
    else return(1);                         /* Return 1 if no match */
}
/* ============ example call to LSearch =================
*/
char          theChar;              /* a keyboard character */
Cell          theCell;
ListHandle    theList;              /* let's assume this is already set up */

    /*... WaitNextEvent returns keyDown and you store character code in
        theChar...
    */

LSetSelect( FALSE, theCell, theList );    /* deselect current selection */
SetPt( &theCell, 0,0 );                   /* start at top of list */

LSearch( &theChar, 1, cmp1stChar, &theCell, theList )
LSetSelect( TRUE, theCell, theList );     /* select cell which matched */
LAutoScroll( theList );                   /* scroll in case it's off screen */
```