Double Linked List

The following example illustrates a C programming technique that is not necessarily Macintosh-specific.

```
// Double Linked List
// This is a simple program that demonstrates how to construct a sorted doubly
// linked list. Elements are placed in their correct place in the list when the
// are inserted


#include <stdio.h>
#include <stdlib.h>

// typedefs for list

typedef struct DblList {
      short i;                          // value to sort on
      struct DblList *next; // pointer to next entity in list
      struct DblList *prev; //  pointer to previous entity in list
      }  *DblListPtr, DblList;


DblListPtr gTheList;                    // global variable

// routine prototypes
void Insert(short);
void PrintList(void);

// this routine takes the  integer argument to insert onto the list
// this routine will insert the integer into the proper place in the
// list, i.e. placing the integers in ascending order
void Insert(short x)
{
      DblListPtr temp, temp2;       // temporary pointers

      // if global list is NULL, then this is first thing to be put into list
      if ( !gTheList ) {
            gTheList = (DblListPtr)malloc(sizeof(DblList));
            if ( !gTheList ) {
                  printf ("Problem allocating list\n");
                  exit(1);
            }
            gTheList->i = x;
            gTheList->prev = gTheList->next = NULL;
      }
      else {
            temp = (DblListPtr)malloc(sizeof(DblList));
            if (!temp) {
                  printf ("Problem allocating List\n");
                  exit(1);
            }
            temp->i = x;
            // search until location in list found
            temp2 = gTheList;
```

```
            while ((temp2->i < x) && temp2->next != NULL )
                    temp2 = temp2->next;

            // if next is NULL, then reached last list node without checking it,
            // thus do comparison and insert value accordingly
            if (temp2->next == NULL) {
            if (temp2->i < x) {
                    temp2->next = temp;
                    temp->prev = temp2;
                    temp->next = NULL;
            }
            else if (temp2->i > x) {
                    temp->prev = temp2->prev;
                    if (temp2->prev)
                            temp2->prev->next = temp;
                            temp2->prev = temp;
                            temp->next = temp2;
                    }
            }
            // otherwise, simply place new value infront of value reached
            else {
                    temp->prev = temp2->prev;
                    if (temp2->prev)
                            temp2->prev->next = temp;
                            temp->next = temp2;
                            temp2->prev = temp;
                    }
            // previous will be NULL, if inserted onto front of list, in
            // this case, must reset gTheList.
            if (!temp->prev){
                    gTheList->prev = temp;
                    gTheList = temp;
            }
    }
}

// simple routine to print out contents of list
void PrintList()
{
    DblListPtr temp = gTheList;

    while (temp) {
            printf ("i = %d\n", temp->i);
            temp = temp->next;
    }
}


main()
{

    // initialize the list
    gTheList = NULL;

    // now insert 5 values in random order to show that it works
    Insert(6);
```

```
        Insert(2);
        Insert(4);
        Insert(3);
        Insert(1);

        // now print out contents of list
        PrintList();
}
```