```c
// Stephen M. Jones // 05/12/2024

// Coda-C example #1:  Array and Void objects.

// Task:  Print lines from <stdin> in sorted order.

static int compare(char**aa,char**bb) { returnstrcmp(*aa,*bb); }

static void sortList(Arraylist) {
    qsort(Array_rawAddress(list), Array_count(list), sizeof(pointer), (pointer)compare);
    }

static void codax01() {
    Arraylist=newO(Array);
    while(1) {
        charaa[1024],*bb=fgets(aa,1000,stdin);if(!bb)break;
        char*str=alocO(strlen(bb)+1); strcpy(str,bb);
        Array_addObject(list,str);freeO(str);
        }
    sortList(list);
    for(intj=0;j<Array_count(list);++j) {
        printf("%s",(char*)Array_subInt(list,j));
        }
    freeO(list);
    }

codax_register(codax01)// test harness setup

// Purpose:  To demonstrate how a dynamic Array can simplify C programming.
```

```
// Coda-C adds the following:

// newO()     - create a new object
// alocO()    - create a classless object
// freeO()    - reduce the retain count for an object and release if zero

// pointer            - C data type aka (void *)
// Array              - C data type
// Array_addObject()  - add an object to an Array
// Array_count()      - number of elements in a Array
// Array_subInt()     - return object in an Array by subscript
// Array_rawAddress() - get base address of array

/*<stdin> test input
a
eeeee
ccc
bb
dddd
**********************/

/*<stdout> example's output
a
bb
ccc
dddd
eeeee
**********************/
```