1) array.h

```
1  #ifndef ARRAY_H
2  #define ARRAY_H
3
4  typedef struct array {
5    void** data;
6    int count;
7    int size;
8  } Array;
9
10 typedef int CompareFunc(void* value1, void* value2);
11 typedef void ActionFunc(void* value, void* param);
12
13 void arInit(Array* a, int size);
14 void arClose(Array* a);
15 void arSetSize(Array* a, int size);
16 void arAdd(Array* a, void* value);
17 void arInsertAt(Array* a, int index, void* value);
18 void arRemoveAt(Array* a, int index);
19 int  arFind(Array *a, void* value, CompareFunc *cmp);
20 void arDoForEach(Array *a, ActionFunc *action, void *param);
21 int  arBinarySearch(Array* a, void* data, CompareFunc *cmp);
22 void arSort(Array* a, CompareFunc *cmp);
23
34 #endif
```

2) array.c

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include "array.h"
4
5   void arInit(Array* a, int size) {
6     a->data = malloc(sizeof(void*) * size);
7     a->count = 0;
8     a->size = size;
9   }
10
11  void arClose(Array* a) {
12    free(a->data);
13  }
14
15  void arSetSize(Array* a, int size) {
16    int newsize = max(size, 2);
17    a->data = (void**)realloc( a->data, sizeof(void*) * size);
18    a->size = newsize;
19    if (a->count > a->size) a->count = a->size;
20  }
21
22  void arAdd(Array* a, void* value) {
23    if (a->count >= a->size)
24      arSetSize(a, a->size * 2);
25    a->data[a->count] = value;
26    ++a->count;
27  }
28
29  void arInsertAt(Array* a, int index, void* value) {
30    int i;
31    if (a->count >= a->size) arSetSize(a, a->size * 2);
32    for (i = a->count-1; i >= index; --i)
33      a->data[i+1] = a->data[i];
34    a->data[index] = value;
35    ++a->count;
36  }
37
38  void arRemoveAt(Array* a, int index) {
39    int i;
40    for (i = index+1; i <= a->count - 1; ++i)
41      a->data[i-1] = a->data[i];
42    --a->count;
43  }
44
45  int arFind(Array *a, void* value, CompareFunc *cmp) {
46    int i;
47    for (i=0; i < a->count; ++i)
48      if (cmp(a->data[i], value) == 0) return i;
49    return -1;
50  }
51
52  void arDoForEach(Array *a, ActionFunc *action, void *param) {
53    int i;
54    for (i=0; i < a->count; ++i)
55      action(a->data[i], param);
56  }
57
58  static int binarySearch(Array* a, void* data, CompareFunc *cmp) {
59    int bottom, top;
60    bottom = 0;
61    top = a->count - 1;
62    while (bottom <= top)
63    {
64      int i, t;
65      i = (top + bottom) / 2;
66      t = cmp(a->data[i],data);
67      if (t == 0) return i;
68      if (t < 0)
```

```
69        bottom = i + 1;
70      else
71        top = i - 1;
72    }
73    return bottom;
74  }
75
76  int arBinarySearch(Array* a, void* data, CompareFunc *compareFunc) {
77    int i;
78
79    if (a->count == 0) return -1;
80    i = arBinarySearch(a, data, compareFunc);
81    if (i >= a->count) return -1;
82    if (compareFunc(a->data[i], data) == 0)
83      return i;
84    return -1;
85  }
86
87  void arQuicksort(Array* a, int lo, int hi, CompareFunc *cmp) {
88    int i=lo, j=hi;
89    void *h, *x = a->data[(lo+hi)/2];
90
91    do {
92      while (cmp(a->data[i], x) < 0) i++;
93      while (cmp(a->data[j], x) > 0) j--;
94      if (i<=j) {
95        h=a->data[i]; a->data[i]=a->data[j]; a->data[j]=h;
96        i++; j--;
97      }
98    } while (i<=j);
99
100   if (lo<j) arQuicksort(a, lo, j, cmp);
101   if (i<hi) arQuicksort(a, i, hi, cmp);
102 }
103
104 void arSort(Array* a, CompareFunc *cmp) {
105   arQuicksort(a, 0, a->count-1, cmp);
106 }
```

3) example1.c

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "array.h"

int compareInteger( void* i1, void* i2) {
  return (int)i1 - (int)i2;
}

int compareString( void* s1, void* s2) {
  return strcmp(s1, s2);
}

void printInteger(void* p, void* param) {
  printf("%d ", p);
}

void printString(void* p, void* param) {
  printf("%s ", p);
}

void main() {
  Array a1, a2;
  int i;
  char s[100];

  arInit(&a1, 10);
  for (i = 0; i < 20; ++i)
      arAdd(&a1, (void*)(rand() % 100));

  printf("sorting integer: ");
  arSort(&a1, compareInteger);
  arDoForEach(&a1, printInteger, NULL);
  printf("Wn");

  arInit(&a2, 10);
  for (i = 0; i < 20; ++i) {
      sprintf(s, "%d", rand() % 100);
      arAdd(&a2, strdup(s));
  }

  printf("sorting string: ");
  arSort(&a2, compareString);
  arDoForEach(&a2, printString, NULL);
  printf("Wn");
}
```