Chapter Twelve

Searching and Sorting

1

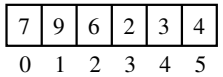
Searching and Sorting

- *Searching* is the process of finding a particular element in an array
- *Sorting* is the process of rearranging the elements in an array so that they are stored in some well-defined order

2

An Example

Find the value 3 in the following array:



3

Searching in an Integer Array

```
int findIntInArray(int key, int array[], int size)
{
   int  i;

   for (i = 0; i < size; i++) {
      if (key == array[i]) return i;
   }
   return –1;
}
```

4

An Example

Given a coin, print the name of the coin:

> 5
nickel
> 25
quarter
> 1
penny

5

Searching Parallel Arrays

0	1
1	5
2	10
3	25
4	50

0	penny
1	nickel
2	dime
3	quarter
4	half-dollar

6

Searching Parallel Arrays

```
static string coinNames[] = {
   “penny”,
   “nickel”,
   “dime”,
   “quarter”,
   “half-dollar”
};

static int coinValues[] = {1, 5, 10, 25, 50};

static int nCoins = sizeof coinValues / sizeof coinValues[0];
```

7

Searching Parallel Arrays

```
main()
{
   int  value, index;
   printf(“Enter coin value:”); scanf(“%d”, &value);
   index = findIntInArray(value, coinValues, nCoins);
   if (index == -1) {
      printf(“There is no such coin.\n”);
   } else {
      printf(“That is called a %s.\n”, coinNames[index]);
   }
}
```

8

An Example

What is the distance between Seattle and Boston?

	Atlanta	Boston	Chicago	Detroit	Houston	Seattle
Atlanta	0	1108	708	732	791	2625
Boston	1108	0	994	799	1830	3016
Chicago	708	994	0	279	1091	2052
Detroit	732	799	279	0	1276	2327
Houston	791	1830	1091	1276	0	2369
Seattle	2625	3016	2052	2327	2369	0

9

Searching in a String Array

```
#define NCities 6
static int mileageTable[NCities][NCities] = {
  { 0,    1108,    708,    732,    791,    2625},
  {1108,    0,     994,    799,   1830,    3016},
  { 708,   994,      0,    279,   1091,    2052},
  { 732,   799,    279,      0,   1276,    2327},
  { 791,  1830,   1091,   1276,      0,    2369},
  {2625,  3016,   2052,   2327,   2369,       0 }
};
static string cityTable[NCities] = {
   "Atlanta", "Boston", "Chicago",
   "Detroit", "Houston", "Seattle"
};
```

10

Searching in a String Array

```
main()
{
   int city1, city2;

   city1 = getCity("Enter name of city #1: ");
   city2 = getCity("Enter name of city #2: ");
   printf("Distance between %s", cityTable[city1]);
   printf(" and %s:", cityTable[city2]);
   printf(" %d miles.\n", mileageTable[city1][city2]);
}
```

11

Searching in a String Array

```
static int getCity(string prompt)
{
   string cityName;
   int index;
   while (TRUE) {
      printf("%s", prompt);
      cityName = getLine();
      index = findStringInArray(cityName, cityTable, NCities);
      if (index >= 0) break;
      printf("Unknown city name – try again.\n");
   }
   return index;
}
```

12

Searching in a String Array

```
int findStringInArray(string key, string array[], int size)
{
   int  i;

   for (i = 0; i < size; i++) {
      if (stringEqual(key, array[i])) return i;
   }
   return –1;
}
```

13

Searching Algorithms

- *Linear search*: the search starts at the beginning of the array and goes straight down the line of elements until it finds a match or reaches the end of the array
- *Binary search*: the search starts at the center of a *sorted* array and determines which half to continue to search on that basis

14

Binary Search

15

Binary Search

```
int binarySearch(string key, string array[], int size) {
   int lh, rh, mid, cmp;
   lh = 0; rh = size –1;
   while (lh <= rh) {
      mid = (lh + rh) / 2;
      cmp = stringCompare(key, array[mid]);
      if (cmp == 0) return mid;
      if (cmp < 0) {
         rh = mid –1;
      } else {
         lh = mid + 1;
      }
   }
   return –1;
}
```

16

Relative Efficiency

- Linear search
  - N comparisons
- Binary search
  - $N/2/2/\ldots/2 = 1$
  - $N = 2^k$
  - $k = \log_2 N$ comparisons

N	$\log_2 N$
10	3
100	7
1,000	10
1000,000	20
1,000,000,000	30

17

Sorting an Integer Array

31	41	59	26	53	58	97	93
26	41	59	31	53	58	97	93
26	31	59	41	53	58	97	93
26	31	41	59	53	58	97	93
26	31	41	53	59	58	97	93
26	31	41	53	58	59	97	93
26	31	41	53	58	59	97	93
26	31	41	53	58	59	93	97

18

Sorting by Selection

```
void sortIntArray(int array[], int size)
{
   int  lh, rh;

   for (lh = 0; lh < size; lh++) {
      rh = findSmallestInt(array, lh, size – 1);
      swap(array, lh, rh);
   }
}
```

19

Sorting by Selection

```
int findSmallestInt(int array[], int low, int high)
{
   int  i, spos;

   spos = low
   for (i = low; i <= high; i++) {
      if (array[i] < array[spos]) spos = i;
   }
   return spos;
}
```

20

Evaluating Performance

N	Running Time
10	0.13
20	0.33
30	1.00
40	1.47
50	2.40
100	9.67
200	37.33
400	146.67
800	596.67

21

Analyzing Performance

The number of comparisons

$= N + (N - 1) + (N - 2) + \ldots + 3 + 2 + 1$

$= (N^2 + N) / 2$

The performance of the selection sort algorithm is *quadratic*

22