Chapter 5. 배열

박 종 혁 교수
UCS Lab
(http://www.parkjonghyuk.net)

Tel: 970-6702
Email: jhpark1@snut.ac.kr

Learning Objectives

- 배열의 소개
  - 배열의 선언과 참조
  - For 루프와 배열
  - 메모리 상의 배열
- 함수에서의 배열
  - 함수 인자로써의 배열과 리턴 값
- 배열 프로그램
  - 부분적으로 채워진 배열, 검색, 정렬
- 다차원 배열

배열 소개

- 배열의 정의:
  - 같은 데이터 형을 가지는 데이터의 집합
- 첫번째 "집합적" 데이터 형
  - 그룹화
  - int, float, double, char 들은 단순 데이터 형
- 아이템의 리스트에 사용
  - 시험 성적, 온도, 이름 등
  - 단순한 변수의 중복된 선언을 피하기 위함
  - 리스트를 하나의 개체 처럼 조작이 가능

배열의 선언

- 배열의 선언 → 메모리 할당
  int score[5];
  - "score"라는 이름의 5개 정수형 배열 선언
  - 5개의 변수를 선언하는 효과:
    int score[0], score[1], score[2], score[3], score[4]
- 배열의 구성요소:
  - 인덱스(Index) 또는 첨자(subscript) 변수
  - 배열의 요소(element)
  - 대괄호 안의 값을 인덱스 또는 첨자라고 한다
    - [0] to [size – 1]

배열에 접근하기

- 인덱스 / 첨자를 이용한 접근
  - cout << score[3];
- 대괄호의 두 가지 사용법:
  - 선언 부분에서는 배열의 크기를 지정
  - 나머지 부분에서는 배열의 인덱스 또는 첨자를 의미
- 크기, 첨자는 반드시 리터럴일 필요는 없음
  - int score[MAX_SCORES];
  - score[n+1] = 99;
    - n이 2이면, score[3]

배열의 사용

- 강력한 저장 매커니즘
- 다음과 같은 명령이 가능:
  - "i번째 인덱스 변수까지 이렇게 실행하라"
    여기에서 i는 프로그램에 의해 계산된다
  - "score 배열의 모든 요소를 출력하라"
  - "사용자로부터 입력받아 score 배열을 채워라"
  - "스코어 배열에서 가장 높은 값을 찾아라"
  - "스코어 배열에서 가장 낮은 값을 찾아라"

디스플레이 5.1 배열을 사용한 프로그램 (1 of 2)

Display 5.1 Program Using an Array

```
1  //Reads in five scores and shows how much each
2  //score differs from the highest score.
3  #include <iostream>
4  using namespace std;
5  int main( )
6  {
7      int i, score[5], max;
8      cout << "Enter 5 scores:\n";
9      cin >> score[0];
10     max = score[0];
11     for (i = 1; i < 5; i++)
12     {
13         cin >> score[i];
14         if (score[i] > max)
15             max = score[i];
16         //max is the largest of the values score[0],..., score[i].
17     }
```

디스플레이 5.1 배열을 사용한 프로그램 (2 of 2)

```
18        cout << "The highest score is " << max << endl
19             << "The scores and their\n"
20             << "differences from the highest are:\n";
21        for (i = 0; i < 5; i++)
22            cout << score[i] << " off by "
23                 << (max - score[i]) << endl;
24        return 0;
25    }
```

SAMPLE DIALOGUE

Enter 5 scores:
5 9 2 10 6
The highest score is 10
The scores and their
differences from the highest are:
5 off by 5
9 off by 1
2 off by 8
10 off by 0
6 off by 4

배열과 for 루프

- 자연 카운팅 루프
  - 배열 요소를 순회하면서 동작
- 예:

```
for (idx = 0; idx<5; idx++)
{
      cout << score[idx] << "off by "
                  << max – score[idx] << endl;
}
```

  - 루프 컨트롤 변수 (idx) 는 0부터 5까지 카운트 됨
    - 루프 바디 부분은 0에서 4까지 실행된다

주요 배열의 함정

- 배열의 인덱스는 항상 0으로 시작!
- 0은 컴퓨터 과학자들에게 첫 번째 숫자
- 배열 범위를 벗어나는 프로그래밍
  - 예측할 수 없는 결과
  - 컴파일러는 이러한 에러를 감지하지 못함!
- 프로그래머가 배열 범위를 벗어나지 않게 해야 한다

주요 배열의 함정의 예

- 인덱스의 범위는 0 부터 array_size – 1 까지
  - 예:

    double temperature[24]; // 24는 배열의 크기
    // "temperature"라는 이름의 24개의 double 형 데이터를 가지는 배열을 선언
    - 배열의 요소는 다음과 같음:
      temperature[0], temperature[1] … temperature[23]
  - 일반적인 실수:

    temperature[24] = 5;
    - 인덱스 24 는 범위를 벗어남!
    - 경고 없이 좋지 않은 결과를 초래한다 – 실행도중 오류 발생

배열의 크기를 정의된 상수로 사용

- 배열의 크기에 정의/명명된 상수를 항상 사용
- 예:

```
const int NUMBER_OF_STUDENTS = 5;
int score[NUMBER_OF_STUDENTS];
```

- 가독성 향상
- 융통성 향상
- 유지보수성 향상

정의된 상수의 사용

- 배열의 크기가 필요할 때 언제든지 사용 가능
  - for 루프에서 :

```
for (idx = 0; idx < NUMBER_OF_STUDENTS; idx++)
{
   // 배열의 조작
}
```

  - 연관된 크기의 계산:

```
lastIndex = (NUMBER_OF_STUDENTS – 1);
```

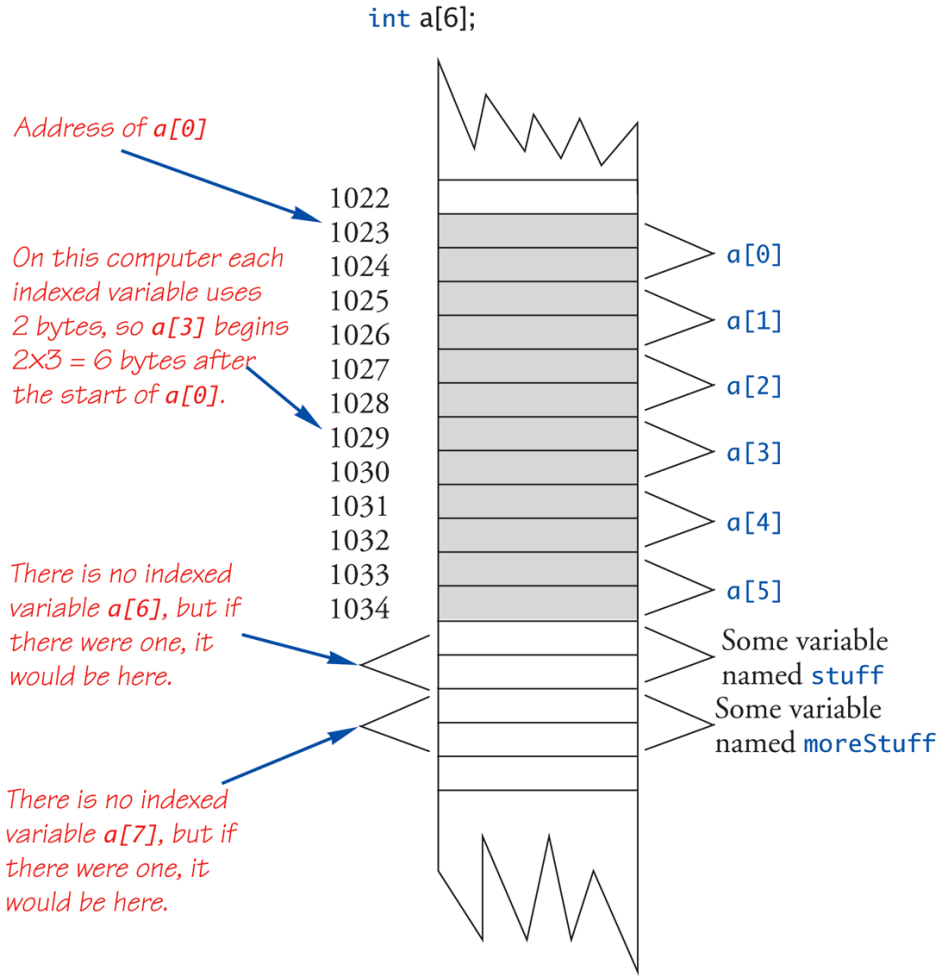
  - 함수로 배열을 넘겨줄 때 (later)
- 크기의 변경 → 프로그램에서 한 부분만 수정하면 된다!

메모리 상의 배열 (1/2)

- 단순 배열을 상기해보면:
  - "주소"를 가지고 메모리에 할당
- 배열 선언은 전체 배열을 메모리에 할당시킴
- 연속적인 할당
  - 주소가 연속적으로 할당된다는 의미
  - 인덱스 계산이 허용됨
    - 배열 시작점 (인덱스 0)으로부터 인덱스를 증가시킴으로써

메모리 상의 배열 (2/2)

Display 5.2 **An Array in Memory**



배열의 초기화

- 단순 변수들이 선언과 동시에 초기화할 수 있는 것과 같이:

  int price = 0; // 0은 초기값

- 배열은 다음과 같이 가능:

  int children[3] = {2, 12, 1};

  – 다음과 같은 의미:

  ```
  int children[3];
  children[0] = 2;
  children[1] = 12;
  children[2] = 1;
  ```

배열의 자동 초기화

- 배열의 크기보다 적은 수의 값이 주어진 경우:
  - 처음부터 순서대로 채워짐
  - 나머지는 배열 기본형의 "0"값으로 채워짐
- 배열의 크기가 없는 경우
  - 초기화 값의 숫자에 따라 배열의 크기 결정
  - 예:

    int b[] = {5, 12, 11};
    - 배열 b의 크기는 3이 된다

함수에서의 배열

- 함수 인자로서의 배열
  - 인덱스 변수
    - 배열의 각각의 요소(element)는 함수의 파라미터로 사용될 수 있다
  - 전체 배열
    - 모든 배열 요소는 하나의 개체로 함수로 전달될 수 있다
- 함수의 리턴값으로서의 배열
  - 가능하다 → chapter 10

인자로서의 인덱스 변수

- 인덱스 변수는 배열 기본 형의 단순 변수처럼 취급된다
- 함수의 선언:
  void myFunction(double par1);
- 변수의 선언:
  int i; double n, a[10];
- 다음과 같은 함수의 호출이 가능:
  myFunction(i); // i는 double형으로 변환
  myFunction(a[3]); // a[3]은 double 형
  myFunction(n); // n은 double 형

인덱스 변수 사용시 주의사항

- myFunction(a[i]);
  - i의 값이 먼저 결정
    - 값이 결정되고 인덱스 변수가 전달됨
- myFunction(a[i*5]);
  - 컴파일러의 관점에서는 완벽하게 정당한 호출
  - 프로그래머는 배열의 범위 내에 들도록 책임져야 함

인자로서의 전체 배열

- 전체 배열은 형식 매개변수가 될 수 있다
  - 함수 호출에서 인자로 전달되는 것은 배열의 이름
  - "배열 매개변수"라 한다
- 배열의 크기도 전달해야 한다
  - 일반적으로 두 번째 매개변수
  - 단순한 int 형의 형식 매개변수

디스플레이 5.3 배열 매개변수를 가지는 함수

Display 5.3 Function with an Array Parameter

SAMPLE DIALOGUEFUNCTION DECLARATION

```
void fillUp(int a[], int size);
//Precondition: size is the declared size of the array a.
//The user will type in size integers.
//Postcondition: The array a is filled with size integers
//from the keyboard.
```

SAMPLE DIALOGUEFUNCTION DEFINITION

```
void fillUp(int a[], int size)
{
    cout << "Enter " << size << " numbers:\n";
    for (int i = 0; i < size; i++)
        cin >> a[i];
    cout << "The last array index used is " << (size - 1) << endl;
}
```

인자로서의 전체 배열의 예

- 주어진 이전의 예:
- main() 함수의 정의에서 이 함수들이 호출되는 경우:

```
int score[5], numberOfScores = 5;
fillup(score, numberOfScores);
```

  - 첫 번째 매개변수는 전체 배열
  - 두 번째 매개변수는 정수형 값
- 배열 매개변수에 대괄호가 없는 것에 주의!

인자로서의 배열 : 어떻게?

- 실제로 무엇이 전달되는가?
- 배열을 3개의 부분으로 나누어 보면
  - 첫 번째 인덱스 변수의 주소 (arrName[0])
  - 배열의 기본 형
  - 배열의 크기
- 단지 첫 번째 부분이 전달됨!
  - 배열 주소의 시작 부분
  - “call-by-reference”와 매우 유사

배열 매개변수

- 익숙하지 않은 형태
  - 배열 매개변수에서 대괄호가 없음
  - 배열의 크기는 별도로 전달이 되어야 한다
- 장점:
  - 단일 함수로 어떠한 크기의 배열도 채울 수 있다!
  - 함수의 "재사용" 특성을 보여주는 예시
  - 예:

```
int score[5], time[10];
fillUp(score, 5);
fillUp(time, 10);
```

const 매개변수 수정자

- 상기할 것 : 배열 매개변수는 실제로 첫 번째 요소의 주소가 전달됨
  - call-by-reference와 유사
- 함수는 배열의 수정이 가능하다!
  - 종종 필요하지만 때로는 필요하지 않음!
- 수정으로부터 배열 내용을 보호
  - 배열 매개변수 앞에 const 수정자를 사용
    - “상수 배열 매개변수”
    - 컴파일러에게 수정이 허가되지 않음을 알려준다

배열을 리턴하는 함수

- 기본형을 리턴하는 것과 같이 배열을 리턴하지 못함
- “포인터”의 사용이 필요하다
- chapter 10에서 논의할 것임

배열 프로그래밍

- 자주 사용되는 유형
  - 부분적으로 채워진 배열
    - 최대 크기가 반드시 선언되어야 한다
  - 정렬
  - 검색(탐색)

부분적으로 채워진 배열

- 필요한 배열 크기의 정확한 예측은 어렵다
- 가능한 가장 큰 크기로 선언되어야 한다
  - 배열에서 입력된 마지막 데이터의 위치를 추적해야 한다
  - 추적 (tracking) 에 필요한 변수가 추가되어야 한다
    - int numberUsed;
    - 배열에서 현재 요소의 숫자를 추적

디스플레이 5.5 부분적으로 채워진 배열 (1 of 5)

Display 5.5 Partially Filled Array

```
1   //Shows the difference between each of a list of golf scores and their average.
2   #include <iostream>
3   using namespace std;
4   const int MAX_NUMBER_SCORES = 10;

5   void fillArray(int a[], int size, int& numberUsed);
6   //Precondition: size is the declared size of the array a.
7   //Postcondition: numberUsed is the number of values stored in a.
8   //a[0] through a[numberUsed-1] have been filled with
9   //nonnegative integers read from the keyboard.

10  double computeAverage(const int a[], int numberUsed);
11  //Precondition: a[0] through a[numberUsed-1] have values; numberUsed > 0.
12  //Returns the average of numbers a[0] through a[numberUsed-1].

13  void showDifference(const int a[], int numberUsed);
14  //Precondition: The first numberUsed indexed variables of a have values.
15  //Postcondition: Gives screen output showing how much each of the first
16  //numberUsed elements of the array a differs from their average.
```

(continued)

디스플레이 5.5 부분적으로 채워진 배열 (2 of 5)

Display 5.5 Partially Filled Array

```
17   int main( )
18   {
19       int score[MAX_NUMBER_SCORES], numberUsed;

20       cout << "This program reads golf scores and shows\n"
21            << "how much each differs from the average.\n";

22       cout << "Enter golf scores:\n";
23      fillArray(score, MAX_NUMBER_SCORES, numberUsed);
24       showDifference(score, numberUsed);

25       return 0;
26   }
```

디스플레이 5.5 부분적으로 채워진 배열 (3 of 5)

```
27  void fillArray(int a[], int size, int& numberUsed)
28  {
29      cout << "Enter up to " << size << " nonnegative whole numbers.\n"
30           << "Mark the end of the list with a negative number.\n";
31      int next, index = 0;
32      cin >> next;
33      while ((next >= 0) && (index < size))
34      {
35          a[index] = next;
36          index++;
37          cin >> next;
38      }
39      numberUsed = index;
40  }
```

디스플레이 5.5 부분적으로 채워진 배열 (4 of 5)

```
41  double computeAverage(const int a[], int numberUsed)
42  {
43      double total = 0;
44      for (int index = 0; index < numberUsed; index++)
45          total = total + a[index];
46      if (numberUsed > 0)
47      {
48          return (total/numberUsed);
49      }
50      else
51      {
52          cout << "ERROR: number of elements is 0 in computeAverage.\n"
53               << "computeAverage returns 0.\n";
54          return 0;
55      }
56  }
```

디스플레이 5.5 부분적으로 채워진 배열 (5 of 5)

Display 5.5 Partially Filled Array

```
57  void showDifference(const int a[], int numberUsed)
58  {
59      double average = computeAverage(a, numberUsed);
60      cout << "Average of the " << numberUsed
61           << " scores = " << average << endl
62           << "The scores are:\n";
63      for (int index = 0; index < numberUsed; index++)
64      cout << a[index] << " differs from average by "
65           << (a[index] - average) << endl;
66  }
```

SAMPLE DIALOGUE

```
This program reads golf scores and shows
how much each differs from the average.
Enter golf scores:
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
69 74 68 -1
Average of the 3 scores = 70.3333
The scores are:
69 differs from average by -1.33333
74 differs from average by 3.66667
68 differs from average by -2.33333
```

전역 상수 vs. 매개변수

- 상수는 일반적으로 전역으로 선언
  - main()함수의 윗부분에 선언
- 함수는 배열 크기 상수를 매개변수로 갖는다
  - 배열 크기를 의미하는 매개변수를 전달해 줄 필요가 없을까?
    - 기술적으로는 yes
  - 왜 우리는 매개변수를 사용하는가?
    - 함수는 대체로 분리된 파일에 정의된다
    - 함수는 다른 프로그램에 의해 재사용될 수 있다!

배열 탐색

- 배열의 전형적인 사용
- 디스플레이 5.6

디스플레이 5.6 배열 탐색 (1 of 4)

Display 5.6 Searching an Array

```
1  //Searches a partially filled array of nonnegative integers.
2  #include <iostream>
3  using namespace std;
4  const int DECLARED_SIZE = 20;

5  void fillArray(int a[], int size, int& numberUsed);
6  //Precondition: size is the declared size of the array a.
7  //Postcondition: numberUsed is the number of values stored in a.
8  //a[0] through a[numberUsed-1] have been filled with
9  //nonnegative integers read from the keyboard.

10 int search(const int a[], int numberUsed, int target);
11 //Precondition: numberUsed is <= the declared size of a.
12 //Also, a[0] through a[numberUsed -1] have values.
13 //Returns the first index such that a[index] == target,
14 //provided there is such an index; otherwise, returns -1.
```

디스플레이 5.6 배열 탐색 (2 of 4)

```
15  int main( )
16  {
17      int arr[DECLARED_SIZE], listSize, target;

18      fillArray(arr, DECLARED_SIZE, listSize);

19      char ans;
20      int result;
21      do
22      {
23          cout << "Enter a number to search for: ";
24          cin >> target;

25          result = search(arr, listSize, target);
26          if (result == -1)
27              cout << target << " is not on the list.\n";
28          else
29              cout << target << " is stored in array position "
30                   << result << endl
31                   << "(Remember: The first position is 0.)\n";
```

디스플레이 5.6 배열 탐색 (3 of 4)

Display 5.6 Searching an Array

```
32          cout << "Search again?(y/n followed by Return): ";
33          cin >> ans;
34      } while ((ans != 'n') && (ans != 'N'));
35      cout << "End of program.\n";
36      return 0;
37  }

38  void fillArray(int a[], int size, int& numberUsed)
39  <The rest of the definition of fillArray is given in Display 5.5>
40  int search(const int a[], int numberUsed, int target)
41  {
42      int index = 0;
43      bool found = false;
44      while ((!found) && (index < numberUsed))
45      if (target == a[index])
46          found = true;
47      else
48          index++;
```

디스플레이 5.6 배열 탐색 (4 of 4)

```
49        if (found)
50            return index;
51        else
52            return -1;
53    }
```

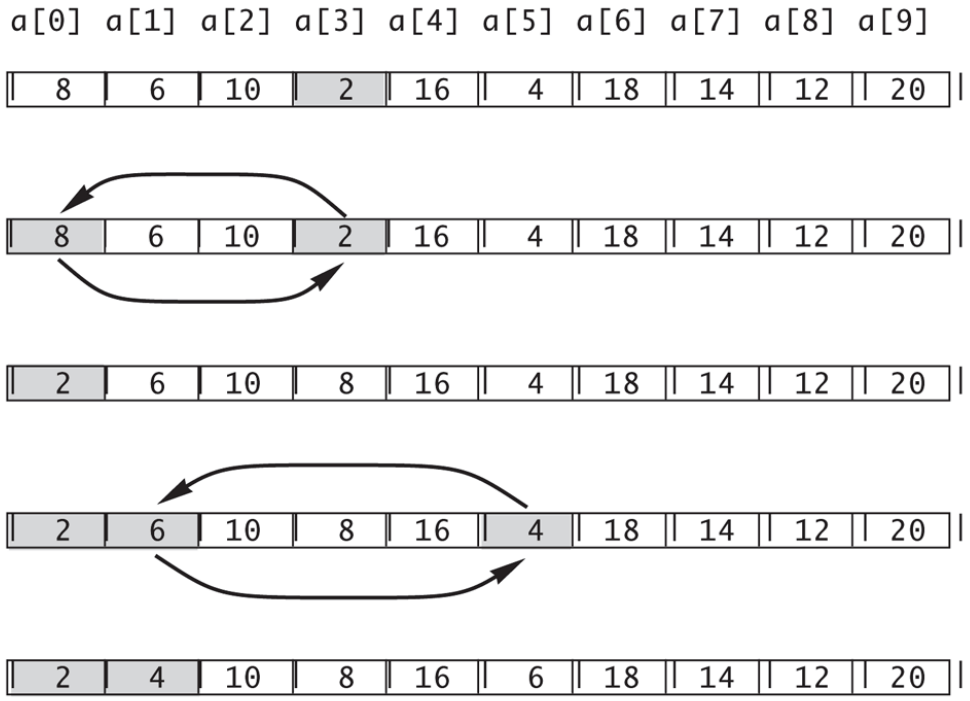
SAMPLE DIALOGUE

```
Enter up to 20 nonnegative whole numbers.
Mark the end of the list with a negative number.
10 20 30 40 50 60 70 80 -1
Enter a number to search for: 10
10 is stored in array position 0
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 40
40 is stored in array position 3
(Remember: The first position is 0.)
Search again?(y/n followed by Return): y
Enter a number to search for: 42
42 is not on the list.
Search again?(y/n followed by Return): n
End of program.
```

디스플레이 5.7 선택 정렬

- 선택 정렬 알고리즘

Display 5.7 Selection Sort



디스플레이 5.8 배열 정렬 (1 of 4)

Display 5.8 Sorting an Array

```
1   //Tests the procedure sort.
2   #include <iostream>
3   using namespace std;

4   void fillArray(int a[], int size, int& numberUsed);
5   //Precondition: size is the declared size of the array a.
6   //Postcondition: numberUsed is the number of values stored in a.
7   //a[0] through a[numberUsed - 1] have been filled with
8   //nonnegative integers read from the keyboard.
9   void sort(int a[], int numberUsed);
10  //Precondition: numberUsed <= declared size of the array a.
```

(continued)

디스플레이 5.8 배열 정렬 (2 of 4)

Display 5.8 Sorting an Array

```
11  //The array elements a[0] through a[numberUsed - 1] have values.
12  //Postcondition: The values of a[0] through a[numberUsed - 1] have
13  //been rearranged so that a[0] <= a[1] <= ... <= a[numberUsed - 1].

14  void swapValues(int& v1, int& v2);
15  //Interchanges the values of v1 and v2.

16  int indexOfSmallest(const int a[], int startIndex, int numberUsed);
17  //Precondition: 0 <= startIndex < numberUsed. Reference array elements
18  //have values. Returns the index i such that a[i] is the smallest of the
19  //values a[startIndex], a[startIndex + 1], ..., a[numberUsed - 1].

20  int main( )
21  {
22      cout << "This program sorts numbers from lowest to highest.\n";

23      int sampleArray[10], numberUsed;
24      fillArray(sampleArray, 10, numberUsed);
25      sort(sampleArray, numberUsed);

26      cout << "In sorted order the numbers are:\n";
27      for (int index = 0; index < numberUsed; index++)
28          cout << sampleArray[index] << " ";
29      cout << endl;

30      return 0;
31  }
```

디스플레이 5.8 배열 정렬 (3 of 4)

```
32  void fillArray(int a[], int size, int& numberUsed)
33            <The rest of the definition of fillArray is given in Display 5.5.>

34  void sort(int a[], int numberUsed)
35  {
36      int indexOfNextSmallest;
37      for (int index = 0; index < numberUsed - 1; index++)
38      {//Place the correct value in a[index]:
39          indexOfNextSmallest =
40                        indexOfSmallest(a, index, numberUsed);
41          swapValues(a[index], a[indexOfNextSmallest]);
42          //a[0] <= a[1] <=...<= a[index] are the smallest of the original array
43          //elements. The rest of the elements are in the remaining positions.
44      }
45  }

46  void swapValues(int& v1, int& v2)
47  {
48      int temp;
49      temp = v1;
50      v1 = v2;
```

디스플레이 5.8 배열 정렬 (4 of 4)

Display 5.8 Sorting an Array

```
51      v2 = temp;
52  }
53
54  int indexOfSmallest(const int a[], int startIndex, int numberUsed)
55  {
56      int min = a[startIndex],
57          indexOfMin = startIndex;
58      for (int index = startIndex + 1; index < numberUsed; index++)
59          if (a[index] < min)
60          {
61              min = a[index];
62              indexOfMin = index;
63              //min is the smallest of a[startIndex] through a[index]
64          }
65      return indexOfMin;
66  }
```

SAMPLE DIALOGUE

```
This program sorts numbers from lowest to highest.
Enter up to 10 nonnegative whole numbers.
Mark the end of the list with a negative number.
80 30 50 70 60 90 20 30 40 -1
In sorted order the numbers are:
20 30 30 40 50 60 70 80 90
```

다차원 배열

- 하나 이상의 인덱스가 추가된 배열
  - char page[30][100];
    - 두 개의 인덱스 : “배열의 배열”
    - 표현:
      page[0][0], page[0][1], …, page[0][99]
      page[1][0], page[1][1], …, page[1][99]
      …
      page[29][0], page[29][1], …, page[29][99]
- C++은 어떠한 수의 인덱스도 허용
  - 일반적으로 두 개 이상 사용하지 않음

다차원 배열 매개변수

- 1차원 배열과 유사
  - 1차원의 크기는 주어지지 않는다
    - 두 번째 매개변수로서 제공됨
  - 2차원의 크기는 주어진다
- 예:

```
void DisplayPage(const char p[][100], int sizeDimension1)
{
        for (int index1=0; index1<sizeDimension1; index1++)
        {
                for (int index2=0; index2 < 100; index2++)
                        cout << p[index1][index2];
                cout << endl;
        }
}
```

요약 1

- 배열은 같은 형 데이터의 집합
- 배열의 인덱스 변수는 다른 단순한 변수와 동일하게 사용된다
- for-루프 N번 반복하면서 배열을 통과
- 프로그래머는 배열의 범위를 초과하지 않게 해야 한다
- 배열 매개변수는 새로운 종류
  - call-by-reference와 유사

요약 2

- 배열 요소는 연속적으로 저장
  - 메모리에 연속적으로 위치
  - 함수에 전달될 때는 단지 첫 번째 배열 요소의 주소가 전달
- 부분적으로 채워진 배열 → tracking 필요
- 상수 배열 매개변수
  - 배열 내용의 수정을 방지
- 다차원 배열
  - 배열의 배열을 생성

Q&A