

Basic Computer Programming

Lecture 7

Electrical & Electronics Engineering
Chung-Ang University

Contents

- Understand the concept of array.
- Understand the declaration and initialization of array.
- Understand the one-dimensional array.
- Understand the relationship between arrays and strings.
- Understand the multi-dimensional array.

The program we will make in this chapter

- Let's store the values in the array and find the minimum value.

```
Microsoft Visual Studio 디버그 콘솔
[ 12 3 19 6 18 8 12 4 1 19 ]
최소값은 1입니다.
```

- Matrix operations

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
Microsoft Visual Studio 디버그 콘솔
2 0 0
0 2 0
0 0 2
```

Array

- A set of variables where you can restore many values at once

```
int s[10];
```

자료형

배열이름



변수

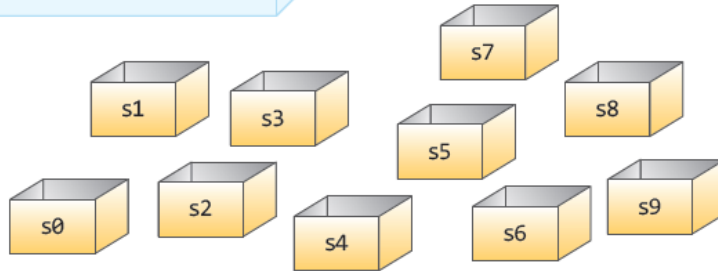


아파트

Why do we need an array?

- Suppose you have 10 students and calculate their average grades.

```
// 일반 변수 사용  
int s0;  
int s1;  
...  
int s9;
```



```
// 배열 사용  
int s[10];
```



Declaration of an array

Syntax 7.1 배열의 선언

형식 자료형 배열이름[배열크기];

예 `int s[10];`

설명 지정된 자료형과 크기를 가지는 배열을 생성한다.

```
int scores[10];
```

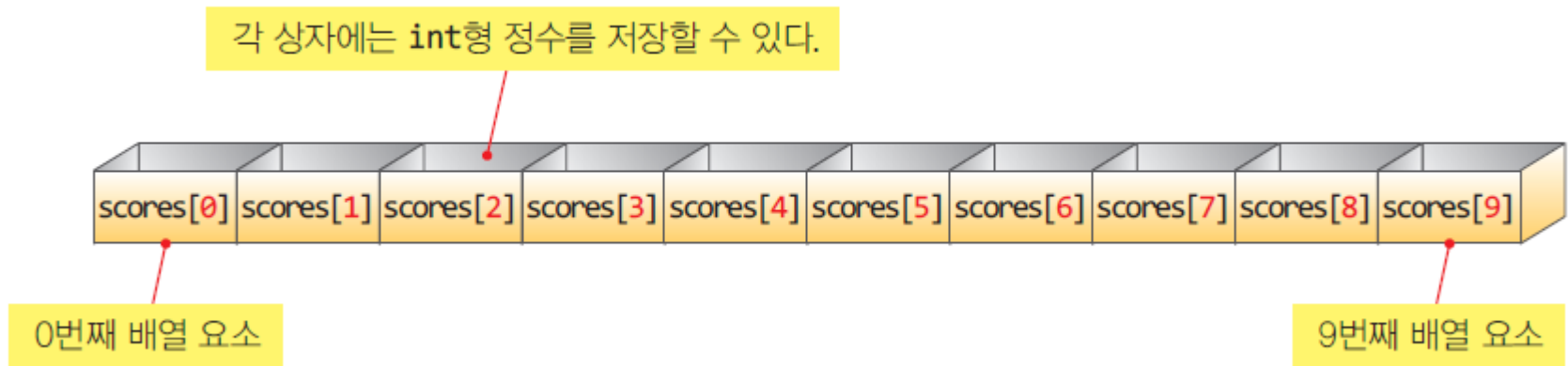
자료형

배열이름

배열크기

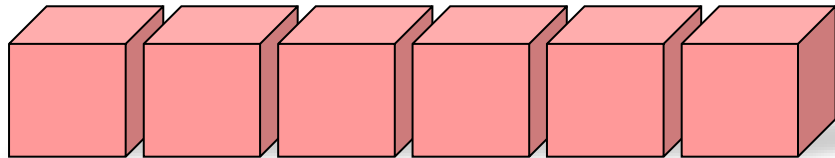
Array elements and indexes

- Index: A number indicating an array element



Example of an array declaration

<code>int prices[60];</code>	<code>// Array with 60 int-type values prices</code>
<code>double costs[12];</code>	<code>// Array with 12 float type values costs</code>
<code>char names[50];</code>	<code>// Array with 50 char type values names</code>



Arrays and iterations

- The advantage of an array is that it is easy to handle the elements of the array using iterative statements



```
scores[0] = 0;  
scores[1] = 0;  
scores[2] = 0;  
scores[3] = 0;  
scores[4] = 0;
```

```
#define SIZE 5  
...  
for(i=0 ; i<SIZE ; i++)  
    scores[i] = 0;
```



Check



참고

배열의 크기를 나타낼 때는 항상 정수 상수를 사용하여야 한다. 변수를 배열의 크기로 사용하면 컴파일 오류가 된다. 또한 배열의 크기를 음수나 0, 실수로 하면 모두 컴파일 오류이다.

```
int scores[size]; // 컴파일 오류
int scores[-2];   // 배열의 크기가 음수이면 안 됨
int scores[6.7];  // 배열의 크기가 실수이면 안 됨
```



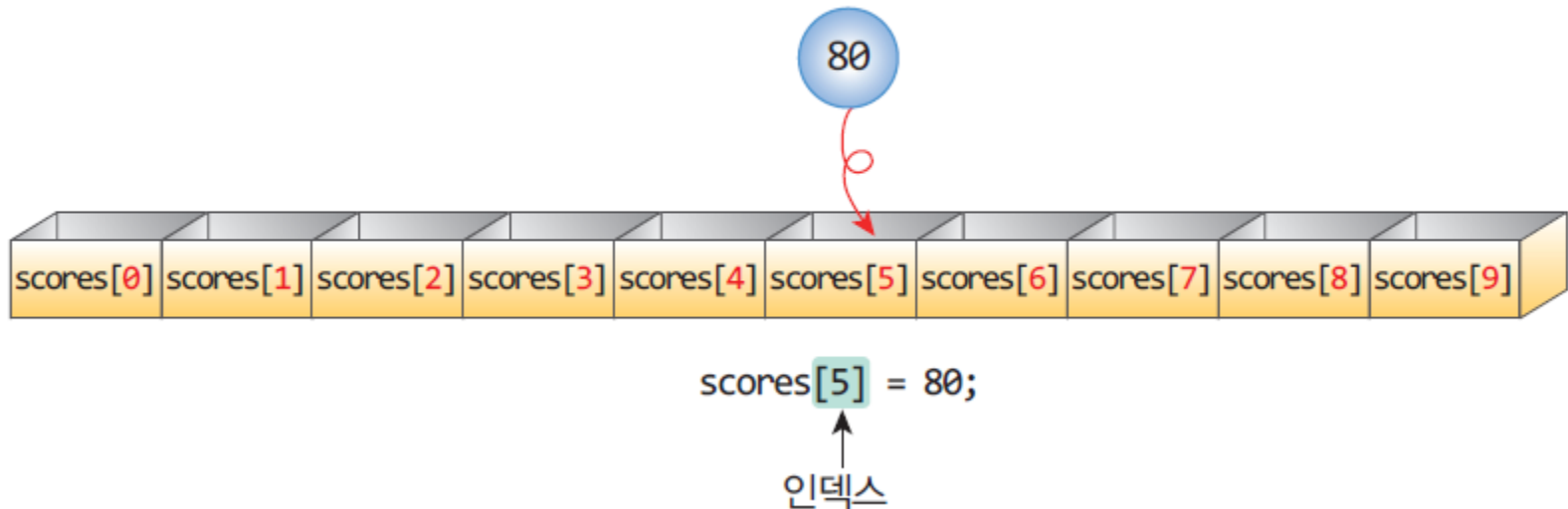
참고

보통 배열을 선언할 때는 배열의 크기를 **#define** 지시자로 만들어진 기호 상수로 지정한다. 예를 들면 다음과 같다.

```
#define SIZE 10
int scores[SIZE];
```

#define을 이용한 기호 상수로 배열의 크기를 지정하게 되면 배열의 크기를 변경하기가 쉬워진다. 즉 프로그램의 다른 부분을 수정하지 않고 단지 기호 상수의 정의만 바꾸면 된다.

Array element access



<code>scores[0] = 80;</code>	// Store 80 in the 0th element.
<code>scores[3] = scores[2];</code>	// Copy the second element to the third element.
<code>scores[k] = 100;</code>	// Store 100 in the kth element.

Example #1

```
#include <stdio.h>

int main(void)
{
    int i;
    int scores[5];

    scores[0] = 10;
    scores[1] = 20;
    scores[2] = 30;
    scores[3] = 40;
    scores[4] = 50;

    for(i=0; i < 5; i++)
        printf("scores[%d]=%d\n", i, scores[i]);
    return 0;
}
```

Microsoft Visual Studio 디버그 콘솔

```
scores[0]=10
scores[1]=20
scores[2]=30
scores[3]=40
scores[4]=50
```

Example #2

```
#include <stdio.h>
#define SIZE 26

int main(void)
{
    int i;
    char codes[SIZE];

    for (i = 0; i < SIZE; i++)
        codes[i] = 'a' + i;           // 'a'에 1을 더하면 'b'가 된다.

    for (i = 0; i < SIZE; i++)
        printf("%c ", codes[i]);
    printf("\n");

    return 0;
}
```



Microsoft Visual Studio 디버그 콘솔

a b c d e f g h i j k l m n o p q r s t u v w x y z

Example #3

```
#include <stdio.h>
#define STUDENTS 5
int main(void)
{
    int scores[STUDENTS];
    int sum = 0;
    int i, average;
    for(i = 0; i < STUDENTS; i++)
    {
        printf("학생들의 성적을 입력하시오: ");
        scanf("%d", &scores[i]);
    }
    for(i = 0; i < STUDENTS; i++)
        sum += scores[i];
    average = sum / STUDENTS;
    printf("성적 평균= %d\n", average);

    return 0;
}
```



학생들의 성적을 입력하시오: 10
학생들의 성적을 입력하시오: 20
학생들의 성적을 입력하시오: 30
학생들의 성적을 입력하시오: 40
학생들의 성적을 입력하시오: 50
성적 평균 = 30

Invalid index problem



경고: 배열 인덱스의 범위

배열을 사용할 때 조심하여야 하는 부분이 배열 인덱스의 범위이다. 인덱스가 배열의 크기를 벗어나게 되면 프로그램에 치명적인 오류를 발생시킨다. 컴파일러는 프로그래머가 유효 범위 안에 있는 인덱스를 사용하고 있는지를 확인하여 주지 않는다. C에서는 프로그래머가 인덱스가 범위를 벗어나지 않았는지를 확인하고 책임을 져야 한다. 예를 들어서 다음의 배열 선언이 있다고 하자.

```
int scores[10];
```

위의 배열에서 사용할 수 있는 인덱스의 범위는 0에서 9까지이다. 다음과 같은 문장은 오류이다. 배열의 인덱스는 0부터 시작한다.

```
scores[10] = 98;
```

Interim check

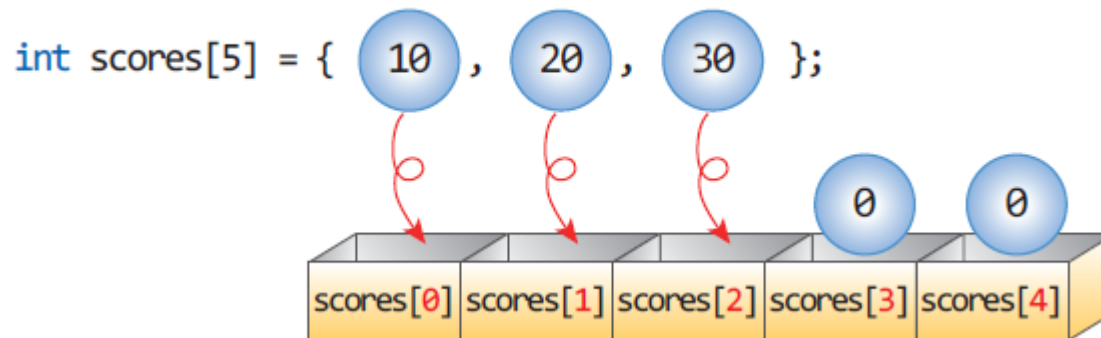
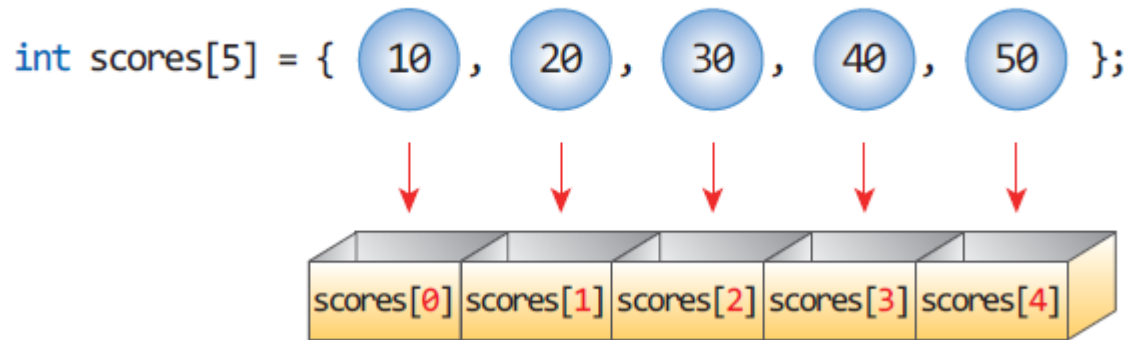


중간점검

1. n 개의 요소를 가지는 배열의 경우, 첫 번째 요소의 번호는 무엇인가?
2. n 개의 요소를 가지는 배열의 경우, 마지막 요소의 번호는 무엇인가?
3. 범위를 벗어나는 인덱스를 사용하면 어떻게 되는가? 즉 `int a[10];`과 같이 선언된 배열이 있는 경우, `a[10]`에 6을 대입하면 어떻게 되는가?

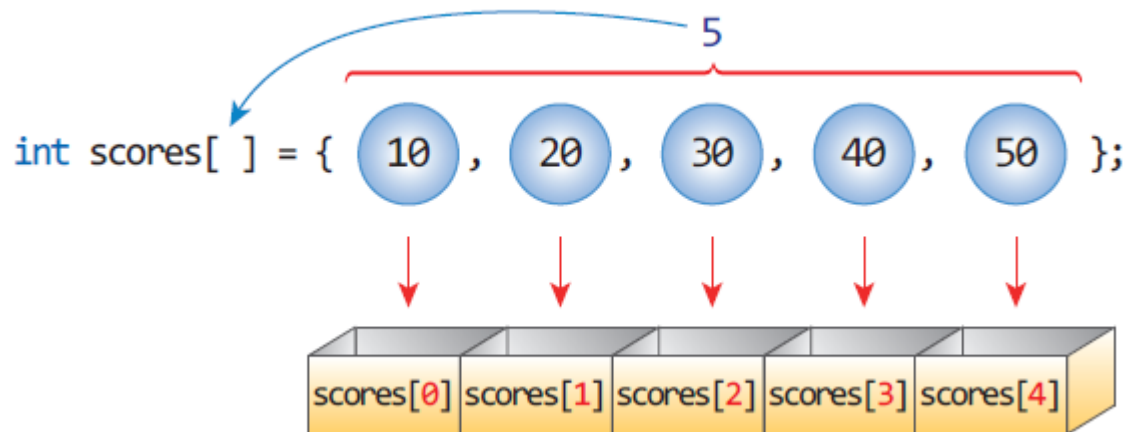


Initialize Array



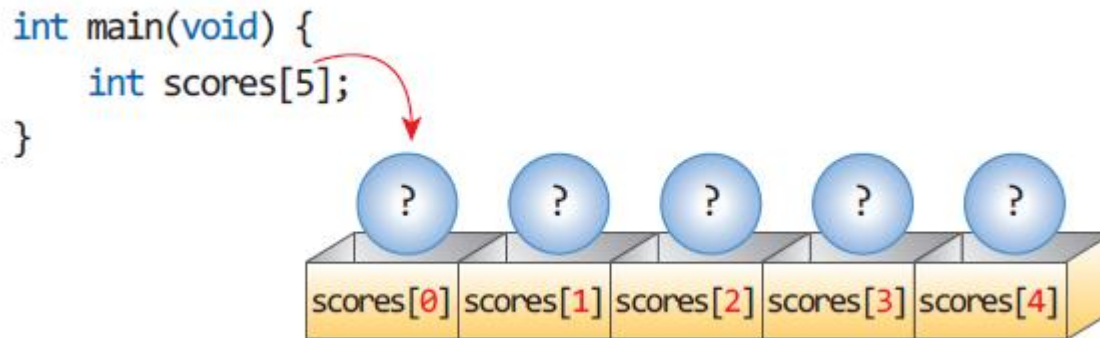
Initialize Array

- If the size of the array is not given, the number of initial values is automatically taken to be the size of the array.



Initialize Array

- If an initial value is not given, a meaningless garbage value is included, just like a general variable.



Example

```
#include <stdio.h>
int main(void)
{
    int scores[5] = { 31, 63, 62, 87, 14 };
    int i;

    for(i = 0; i < 5; i++)
        printf("scores[%d] = %d\n", i, scores[i]);

    return 0;
}
```



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar is orange and contains the text "Microsoft Visual Studio 디버그 콘솔" along with standard window control buttons. The console area is white and displays the output of the program, which is the scores array printed line by line. A vertical scrollbar is visible on the right side of the console.

```
Microsoft Visual Studio 디버그 콘솔
scores[0] = 31
scores[1] = 63
scores[2] = 62
scores[3] = 87
scores[4] = 14
```

Tip



Tip: 배열 요소의 개수를 계산하는 방법

배열에 들어있는 자료를 처리하려면 항상 배열의 처음부터 끝까지 반복하여야 하는 경우가 많다. 따라서 배열의 크기는 꼭 알아야 하는 정보이다. 만약 배열의 크기를 명시적으로 지정하지 않고 주어진 초기값의 개수로 결정하는 경우, 초기값의 개수를 매번 세어보아야 한다. 예를 들면 아래의 문장에서 `scores[]`의 크기는 비교적 쉽게 알 수 있지만 만약 초기값의 개수가 많아지게 되면 정확한 개수를 센다는 것이 어려울 수 있다.

```
int scores[] = { 10, 9, 5, 4, 1, 11, 21, 33, 98, 35, 63, 71 };
```

배열 안에 들어 있는 요소의 개수를 자동적으로 계산하는 방법이 있다. 바로 `sizeof` 연산자를 사용하는 것이다. 우리가 알다시피 `sizeof` 연산자는 자료형이나 변수의 크기를 바이트 단위로 계산하는 연산자이다. `sizeof` 연산자를 이용하여 배열 전체의 크기를 구하고 이것을 배열 요소의 크기로 나누게 되면 배열 요소가 몇 개나 있는지 쉽게 계산할 수 있다.

```
size = sizeof(scores) / sizeof(scores[0]);
```

Interim check

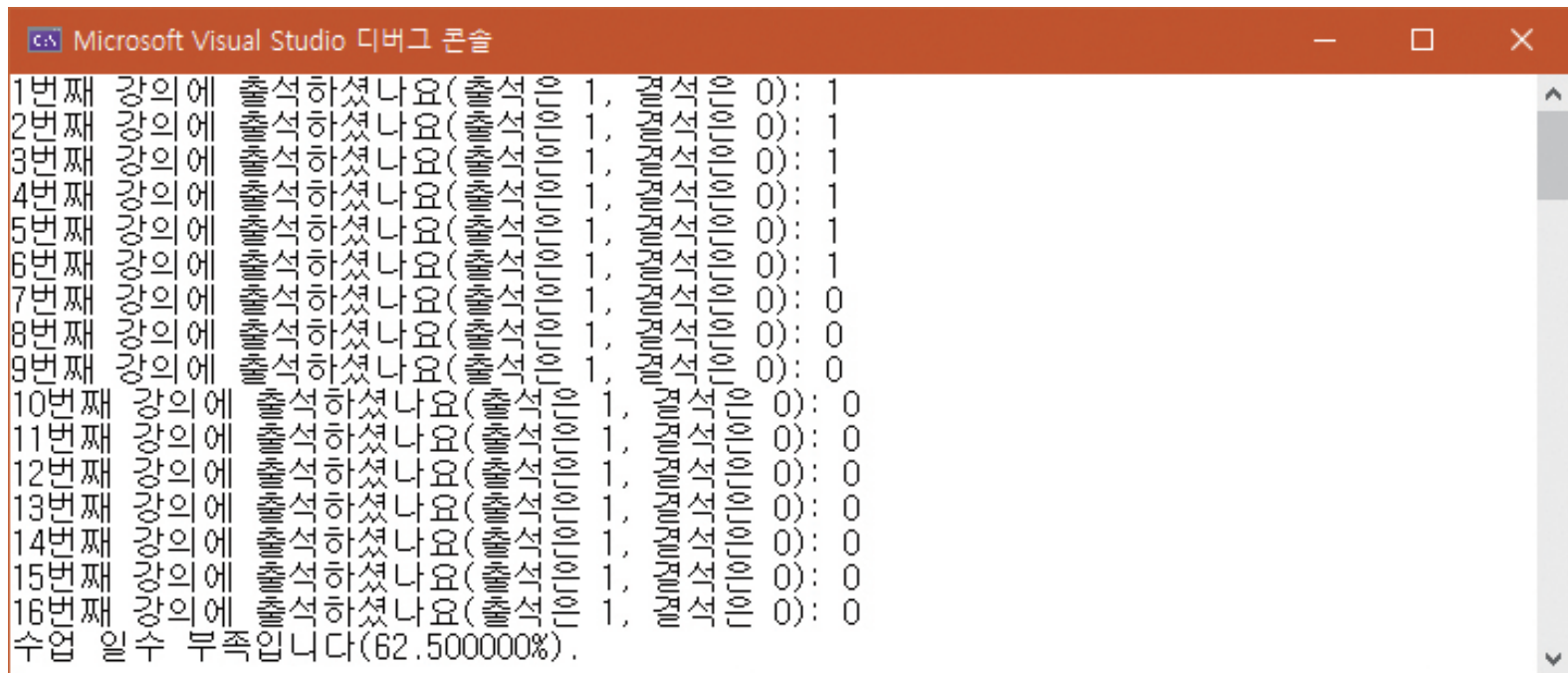


중간점검

1. 배열 `a[6]`의 요소를 1, 2, 3, 4, 5, 6으로 초기화하는 문장을 작성하라.
2. 배열의 초기화에서 초기값이 개수가 배열 요소의 개수보다 적은 경우에는 어떻게 되는가? 또 반대로 많은 경우에는 어떻게 되는가?
3. 배열의 크기를 주지 않고 초기값의 개수로 배열의 크기를 결정할 수 있는가?

Lab: Recording attendance in an array

- Let's implement a simple electronic attendance book using an array



```
Microsoft Visual Studio 디버그 콘솔
1번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 1
2번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 1
3번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 1
4번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 1
5번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 1
6번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 1
7번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
8번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
9번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
10번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
11번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
12번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
13번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
14번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
15번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
16번째 강의에 출석하셨습니다(출석 이력: 1, 현재 출석 이력: 0): 0
수업 일수 부족입니다(62.500000%).
```

Sol:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define SIZE 16

int main(void)
{
    int att_book[SIZE] = { 0 };
    int i, count = 0;

    // 사용자로부터 출석인지 결석인지를 받아서 배열에 저장한다.
    for (i = 0; i < SIZE; i++) {
        printf("%d번째 강의에 출석하셨나요(출석은 1, 결석은 0): ", i + 1);
        scanf("%d", &att_book[i]);
    }
}
```


Sol:

```
// 배열을 검사하여서 결석한 횟수를 계산한다.  
for (i = 0; i < SIZE; i++) {  
    if (att_book[i] == 0)  
        count++;  
}  
  
// 이번 학기 결석률을 계산한다.  
double ratio = count / 16.0;  
if (ratio > 0.3)  
    printf("수업 일수 부족입니다(%f%%). \n", ratio * 100.0);  
  
return 0;  
}
```






Lab: Find the cheapest item

- When we buy products on the Internet, we search for the cheapest places through price comparison sites.
- In general, it is the same as finding the minimum value among integers in an array.



Results

```
Microsoft Visual Studio 디버그 콘솔
[ 12 3 19 6 18 8 12 4 1 19 ]
최소값은 1입니다.
```

Store	Certified rating	Inventory	Price	Total price
 Your Trusted Source since 1983	★★★★★ Rate this store See store profile	In stock Great Accessory Prices	Price: \$312.00 Tax: \$0.00 Shipping: Free	\$312.00 Your best price Shop now
	★★★★★ Rate this store See store profile	In stock	Price: \$312.95 Tax: \$0.00 Shipping: Free	\$312.95 Shop now
	★★★★★ Rate this store See store profile	In stock	Price: \$312.95 Tax: \$0.00 Shipping: Free	\$312.95 Shop now
	★★★★★ Rate this store See store profile	In stock	Price: \$313.00 Tax: \$0.00 Shipping: Free	\$313.00 Shop now
	Not yet rated Rate this store See store profile	In stock	Price: \$316.50 Tax: \$0.00 Shipping: Free	\$316.50 Shop now

Algorithm

1. *Initialize the element of array prices[] to a random number.*
2. *First, assume that the first element is the minimum value of minimum.*
3. *for(i=1; i<sizeofArray; i++)*
4. *if (prices[i] < minimum)*
5. *minimum = prices[i]*
6. *At the end of the iteration, the minimum is stored.*

Sol.

```
#include <stdio.h>
#define SIZE 10

int main(void)
{
    int prices[SIZE] = { 12, 3, 19, 6, 18, 8, 12, 4, 1, 19 };
    int i, minimum;

    printf("[ ");
    for (i = 0; i < SIZE; i++) {
        printf("%d ", prices[i]);
    }
    printf("]\n");

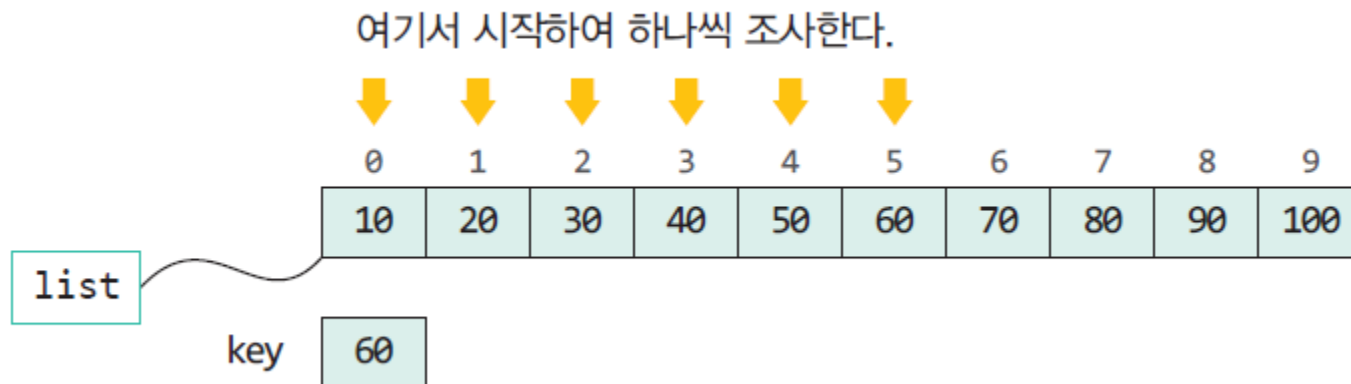
    minimum = prices[0];
    for (i = 1; i < SIZE; i++) {
        if (prices[i] < minimum)
            minimum = prices[i];
    }
    printf("최소값은 %d입니다.\n", minimum);
    return 0;
}
```

Lab: exploring specific values in an array

- Here, it is assumed that integers are stored in an array, where the user finds a specific integer.



```
C:\Windows\system32\cmd.exe  
[ 10 20 30 40 50 60 70 80 90 100 ]  
탐색할 값을 입력하시오:60  
탐색 성공 인덱스= 5  
계속하려면 아무 키나 누르십시오 . . .
```



Sol:

```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
#define SIZE 10

int main(void)
{
    int key, i;

    int list[SIZE] = { 10, 20, 30, 40, 50, 60, 70, 80, 90, 100 };

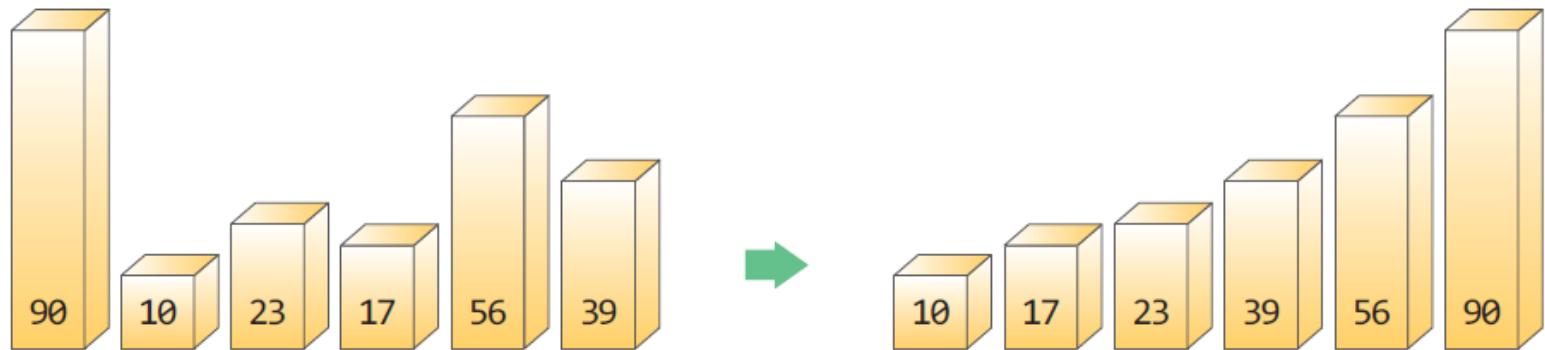
    printf("[ ");
    for (i = 0; i < SIZE; i++) {
        printf("%d ", list[i]);
    }
    printf("]\n");
}
```

Sol:

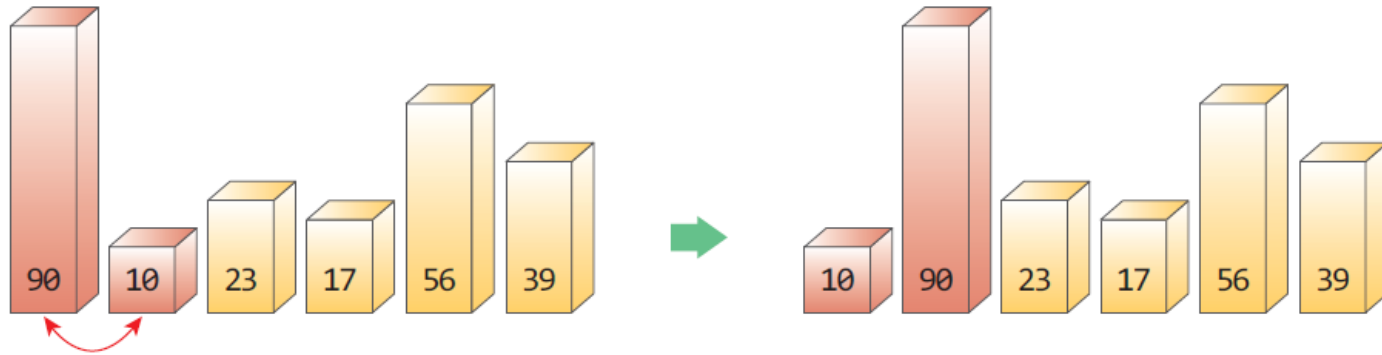
```
printf("탐색할 값을 입력하시오:");  
scanf("%d", &key);  
  
for (i = 0; i < SIZE; i++) {  
    if (list[i] == key) {  
        printf("탐색 성공 인덱스= %d\n", i);  
        break;  
    }  
}  
  
return 0;  
}
```


Sort

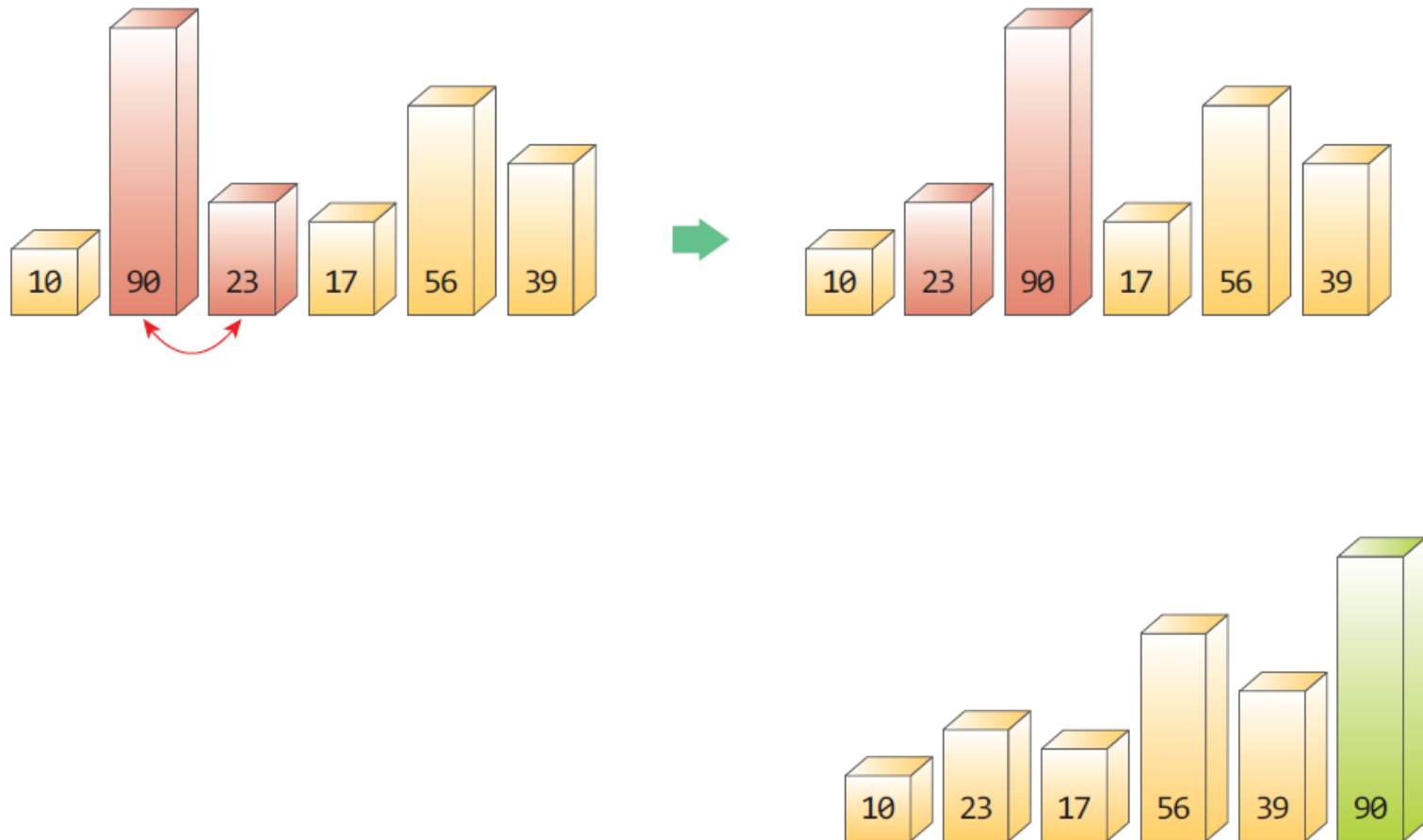
- Sort is the order of things in ascending or descending order by size
- Sort is one of the most basic and important algorithms in computer science



Bubble sort



Bubble sort



Bubble sort

```
#include <stdio.h>
#define SIZE 5

int main(void)
{
    int i, k;
    int list[SIZE] = { 16, 7, 9, 1, 3 };

    // 배열의 요소를 정렬한다.
    for (k = 0; k < SIZE; k++) {
        for (i = 0; i < SIZE - 1; i++) {
            if (list[i] > list[i + 1]) { // 크기 순이 아니면
                // 서로 교환한다.
                int tmp = list[i];
                list[i] = list[i + 1];
                list[i + 1] = tmp;
            }
        }
    }
}
```

Bubble sort

```
// 배열의 요소를 출력한다.  
for (i = 0; i < SIZE; i++) {  
    printf("%d ", list[i]);  
}  
return 0;  
}
```

Interim check



도전문제

본문에서는 버블 정렬을 쉽게 설명하기 위하여 안쪽 **for** 루프를 (**SIZE-1**)번 반복하였다. 하지만 자료 구조와 알고리즘 책에서는 안쪽 **for** 루프가 다음과 같이 기술된다. 이것이 가능한 이유는 무엇인가?

```
for (k = 0; k < SIZE; k++) {  
    for (i = 0; i < SIZE-k-1; i++) {  
        if (list[i] > list[i + 1]) {  
            int tmp = list[i]; list[i] = list[i + 1]; list[i + 1] = tmp;  
        }  
    }  
}
```

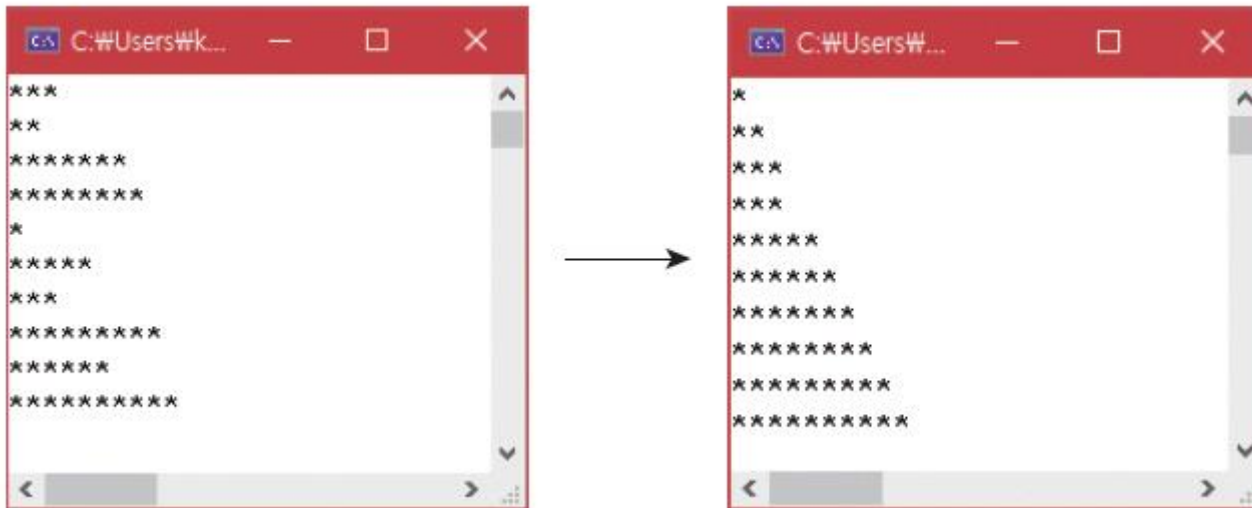


도전문제

버블 정렬의 하나의 패스에서 한 번도 교환이 없으면 정렬이 완료된 것으로 볼 수 있다. 이 부분을 본문의 코드에 추가하여 코드를 업그레이드 해보자.

Lab: Bubble sort with a figure

- Let's write a program that illustrates the process of bubble sort as follows.



Sc

```
#include <windows.h>
#include <stdio.h>
#define SIZE 10

int main(void) {
    int list[SIZE] = { 100, 30, 20, 78, 89, 12, 56, 38, 99, 66 };

    for (int k = 0; k < SIZE; k++) {
        system("cls");    // 화면을 지운다.
        for (int i = 0; i < SIZE - 1; i++) {    // 버블 정렬
            if (list[i] > list[i + 1]) {
                int tmp = list[i]; list[i] = list[i + 1]; list[i + 1] = tmp;
            }
        }
        for (int i = 0; i < SIZE; i++) {
            for (int m = 0; m < list[i] / 10; m++) // 세로로 막대를 그린다.
                printf("*");
            printf("\n");
        }
        _getch();    // 사용자로부터 하나의 문자를 받을 때까지 기다린다.
    }
    return 0;
}
```


Multi-dimensional array

- A multi-dimensional array may have an array element in multiple dimensions. For multi-dimensional arrays, n-dimensional arrays are generally possible, such as two-dimensional arrays and three-dimensional arrays

```
int s[10];           // one dimension  
int s[3][10];        // two dimension  
int s[5][3][10];     // three dimension
```



Two-dimensional array

```
int s[3][10];  // two dimension
```

	열 #0	열 #1	열 #2	열 #3	열 #4
행 #0	s[0][0]	s[0][1]	s[0][2]	s[0][3]	s[0][4]
행 #1	s[1][0]	s[1][1]	s[1][2]	s[1][3]	s[1][4]
행 #2	s[2][0]	s[2][1]	s[2][2]	s[2][3]	s[2][4]

Initialization

```
int s[3][5] = {  
    { 0, 1, 2, 3, 4 }, // 첫 번째 행의 원소들의 초기값  
    { 10, 11, 12, 13, 14 }, // 두 번째 행의 원소들의 초기값  
    { 20, 21, 22, 23, 24 } // 세 번째 행의 원소들의 초기값  
};
```

	열 #0	열 #1	열 #2	열 #3	열 #4
행 #0	0	1	2	3	4
행 #1	10	11	12	13	14
행 #2	20	21	22	23	24

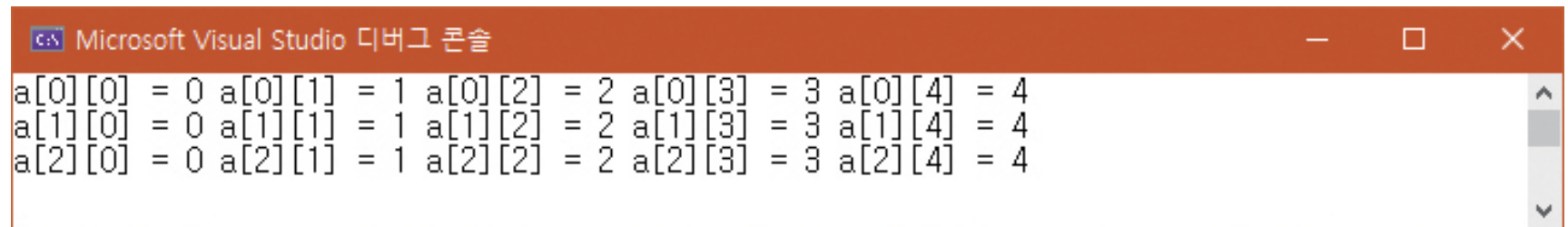
Example #1

```
#include <stdio.h>

int main(void)
{
    int i, j;
    // 3행과 5열을 가지는 2차원 배열 선언
    int a[3][5] = { { 0, 1, 2, 3, 4 }, { 0, 1, 2, 3, 4 }, { 0, 1, 2, 3, 4 } };

    // 각 배열 요소의 값을 출력한다.
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 5; j++) {
            printf("a[%d][%d] = %d ", i, j, a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Results



The screenshot shows the 'Debug Console' window in Microsoft Visual Studio. The title bar is orange and contains the text 'Microsoft Visual Studio 디버그 콘솔' along with standard window control buttons (minimize, maximize, close). The console area is white and displays three lines of text, each representing a row of a 3x5 array 'a'. The values are as follows:

Row	Column 0	Column 1	Column 2	Column 3	Column 4
a[0]	0	1	2	3	4
a[1]	0	1	2	3	4
a[2]	0	1	2	3	4

On the right side of the console, there is a vertical scrollbar with up and down arrow buttons.

Interim check



중간점검

1. 다차원 배열 `int a[3][2]`에는 몇 개의 요소가 존재하는가?
2. 다차원 배열 `int a[3][2]`의 모든 요소를 `0`으로 초기화하는 문장을 작성하시오.

Lab: Matrix

- A matrix is used in natural science to solve many problems

$$A = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad B = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$



A screenshot of a Microsoft Visual Studio debug console window. The title bar is orange and contains the text "Microsoft Visual Studio 디버그 콘솔" (Microsoft Visual Studio Debug Console) and standard window control buttons (minimize, maximize, close). The console area is white and displays a 3x3 matrix of numbers: 2 0 0, 0 2 0, and 0 0 2, arranged in three lines. A vertical scrollbar is visible on the right side of the console area.

```
2 0 0
0 2 0
0 0 2
```

Sol.

```
#include <stdio.h>
#define ROWS 3
#define COLS 3

int main(void)
{
    int r, c;

    int A[ROWS][COLS] = { { 1,0,0 }, { 0,1,0 }, { 0,0,1 } };
    int B[ROWS][COLS] = { { 1,0,0 }, { 0,1,0 }, { 0,0,1 } };
    int C[ROWS][COLS];

    // 두개의 행렬을 더한다.
    for (r = 0; r < ROWS; r++) {
        for (c = 0; c < COLS; c++) {
            C[r][c] = A[r][c] + B[r][c];
            printf("%d ", C[r][c]);
        }
        printf("\n");
    }

    return 0;
}
```