컴퓨터 프로그래밍 및 실습

8주차. 배열

실습 안내

- 실습 제출 안내
  - 솔루션 이름은 “Practice week 8”
  - 프로젝트 이름과 소스코드 이름은 Problem1, Problem2, …
    - 실습1의 프로젝트 이름은 Problem1, 소스코드 이름은 problem1.c
    - 실습 2의 프로젝트 이름은 Problem2, 소스코드 이름은 problem2.c …
  - 솔루션 폴더를 압축하여 **Practice_week8_학번_이름.zip** 으로 제출
  - 제출기한: 당일 **19**시 까지

1. 배열

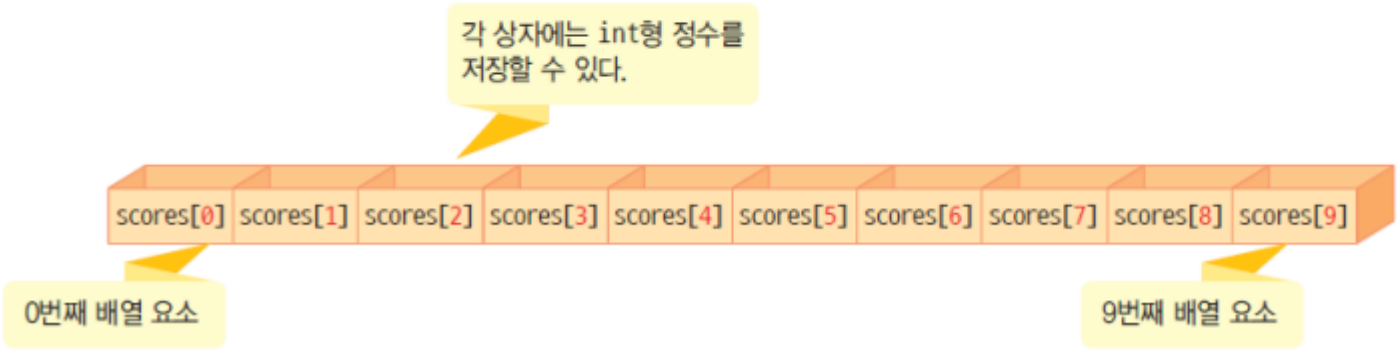
- 배열(array)이란 동일한 타입의 데이터가 여러 개 저장될 수 있는 데이터 저장장소
- 배열 선언

Syntax: 배열 선언

예 `int scores[10];`

- `int` : 자료형
- `scores` : 배열이름
- `10` : 요소의 개수

- 배열 요소와 인덱스(index)

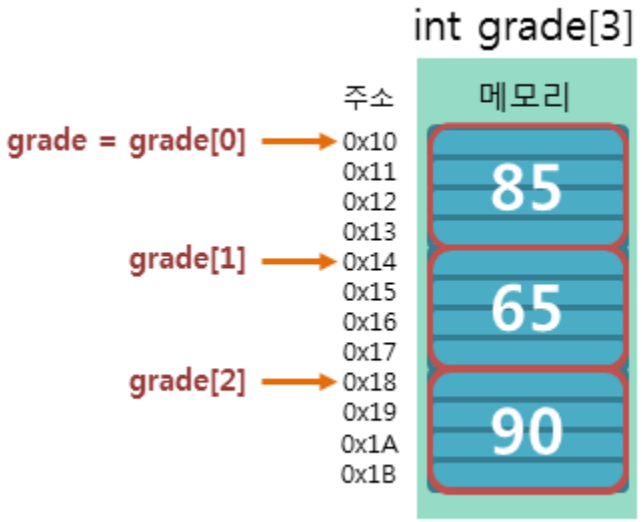


1. 배열

- 배열과 메모리
  - 배열은 메모리 상에서 연속적으로 공간이 할당됨

  - 오른쪽 그림의 경우
    - int grade[3] = { 85, 65, 90 };

    - int 타입은 4byte의 공간을 차지
    - 따라서 총 배열은 4 * 3 = 12 byte의 공간을 차지



1. 배열

```c
#include <stdio.h>

int main(void) {
    int a;  // 변수의 선언
    a = 5;  // 변수의 초기화

    int b = 4;  // 변수의 선언과 초기화를 동시에


    int arr[10];    // 배열의 선언
    arr[0] = 0;     // 배열의 0번째 초기화

    int foo[10] = { 4 };    // 배열의 0번째만 4가 되고, 나머지는 0으로 초기화 된다.

    int score[] = { 100, 45, 87, 60 }; // 자동으로 길이 4의 배열이 선언 및 초기화 된다.

    for (int i = 0; i < sizeof(score) / sizeof(int); i++) {
        printf("%d\n", score[i]);
    }

    return 0;
}
```

1. 배열

- 예제
  - 배열에 성적을 저장한다.
  - 배열에 저장된 성적 중에서 가장 높은
  - 성적을 찾아서 출력

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

#define SCORE_LENGTH 100

int main(void)
{
    int score[SCORE_LENGTH] = { 0 };
    int length;

    printf("입력할 성적의 개수: ");
    scanf("%d\n", &length);

    for (int i = 0; i < length; i++) {
        scanf("%d", &score[i]);
    }

    int max_score = score[0];

    for (int i = 1; i < length; i++) {
        if (max_score < score[i]) {
            max_score = score[i];
        }
    }

    printf("%d\n", max_score);

    return 0;
}
```

1. 배열

- 실습 1
  - 사용자로부터 정수 5개를 입력 받고, 입력 받은 순서의 반대로 출력해보자

    - 예시)
    - 입력: 1 2 3 4 5
    - 출력: 5 4 3 2 1

    - 입력: 5 3 7 9 2
    - 출력: 2 9 7 3 5

1. 배열

- 실습 2
  - 재혁이가 강의 노트의 일부분을 잃어버렸다. 잃어버린 페이지만 다시 프린트하려고 하는데, 어느 부분이 없어졌는지 파악하기 쉽지가 않다. 재혁이를 도와서 잃어버린 페이지를 찾아보자.

  - 정수 A와 B가 입력된다.
    - A페이지와 B페이지 사이에 있는 페이지를 여러 개 잃어버렸다는 뜻.
  - A와 B 사이의 정수가 계속해서 입력된다. (-1을 입력하여 입력을 끝낼 수 있음)
    - 재혁이가 가지고 있는 페이지를 계속해서 입력

  - 재혁이가 잃어버린 페이지를 출력해보자.

  - 예시
    - A=5 B=13
    - 가지고 있는 페이지: 5, 13, 7, 6, 9, 10, 11, -1
    - 출력(잃어버린 페이지): 8, 12

2. 배열과 함수

- 함수의 파라미터로 배열
  - int, double 등은 인자의 사본이
  - 전달된다.
  - 배열은 원본이 그대로 전달된다.

  - Call by value
  - VS
  - Call by reference

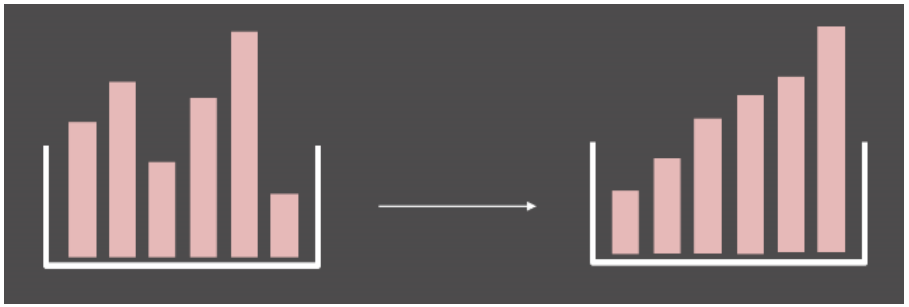
  - 다음 시간에 배울 내용
  - - 배열 그 자체가 포인터이기 때문

  - sizeof에 관한 설명
    - Sizeof 연산자는 컴파일 타임에 치환
    - 따라서 다른 함수에서 배열의 길이를
    - 알 수 없음

```c
#include <stdio.h>

int get_half_age(int age);
void get_half_age_arr(int age_arr[], int len);

int main(void)
{
    int age = 22;

    // get_half_age를 계속 호출해도 age 변수는 변하지 않는다.
    printf("%d\n", get_half_age(age));
    printf("%d\n", get_half_age(age));

    int age_arr[6] = { 10, 22, 50, 25, 30, 21 };
    // get_half_age_arr를 계속 호출하면, age_arr의 요소가 계속 2로 나누어짐
    get_half_age_arr(age_arr, 6);
    for (int i = 0; i < 6; i++) {
        printf("%d ", age_arr[i]);
    }
    printf("\n");

    get_half_age_arr(age_arr, 6);
    for (int i = 0; i < 6; i++) {
        printf("%d ", age_arr[i]);
    }
    printf("\n");

    return 0;
}

int get_half_age(int age)
{
    age /= 2;

    return age;
}

void get_half_age_arr(int age_arr[], int len)
{
    for (int i = 0; i < len; i++) {
        age_arr[i] /= 2;
    }
}
```

3. 정렬

- 정렬 문제
  - 배열에 저장된 요소가 순서가 없이 뒤죽박죽일 때,
  - 배열의 요소를 정렬할 수 있다.

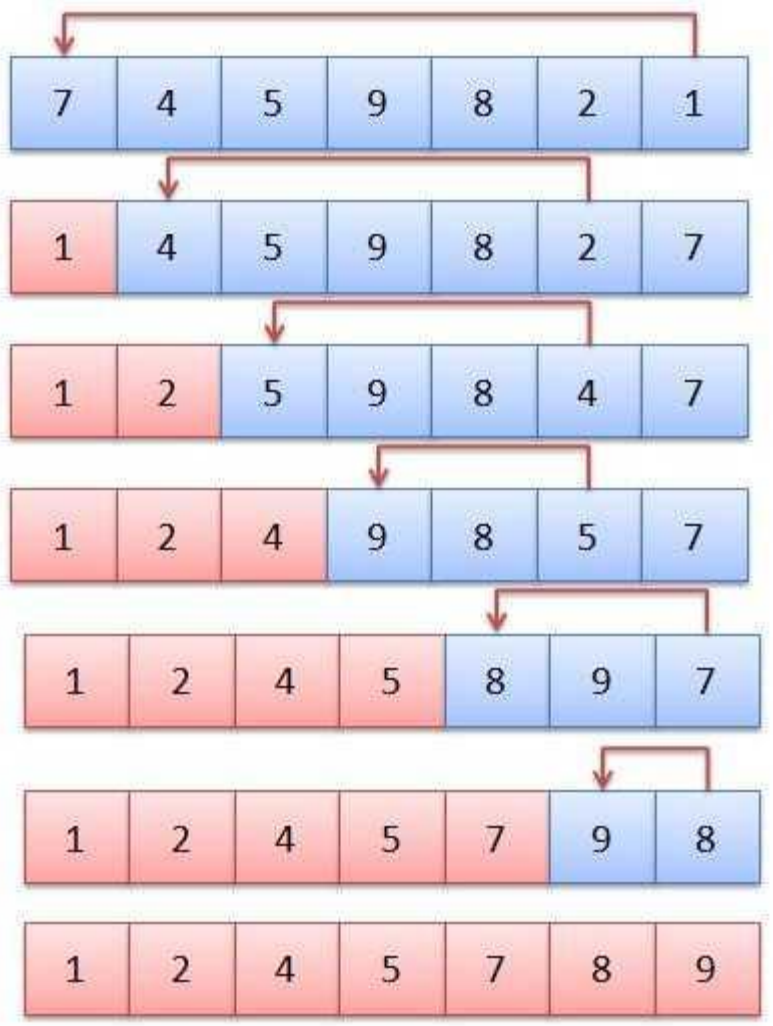


이름	최선	평균	최악	메모리	안정
Bubble sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	O
Insertion sort	$O(n)$	$O(n^2)$	$O(n^2)$	$O(1)$	O
Heap sort	$O(n)$	$O(n \log n)$	$O(n \log n)$	$O(1)$	X
Merge sort	$O(n \log n)$	$O(n \log n)$	$O(n \log n)$	$O(n)$	O
Quick sort	$O(n \log n)$	$O(n \log n)$	$O(n^2)$	$O(\log n)$	X

3. 정렬

- 선택 정렬 (Selection sort)

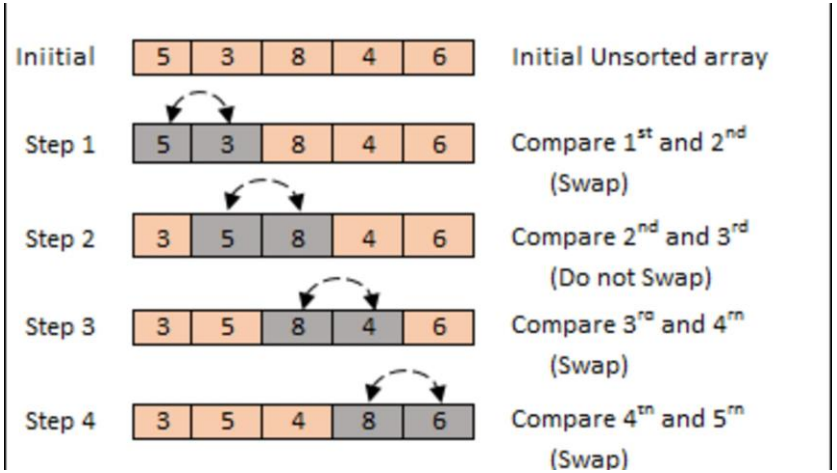
```c
#include <stdio.h>
#define SIZE 10
int main(void)
{
    int list[SIZE] = { 3, 2, 9, 7, 1, 4, 8, 0, 6, 5 };
    int i, j, temp, least;
    for (i = 0; i < SIZE - 1; i++) {
        least = i;
        for (j = i + 1; j < SIZE; j++)
            if (list[j] < list[least])
                least = j;
        temp = list[i];
        list[i] = list[least];
        list[least] = temp;
    }
    for (i = 0; i < SIZE; i++)
        printf("%d ", list[i]);
    printf("\n");
    return 0;
}
```



3. 정렬

- 거품 정렬 (Bubble Sort)

```c
#include <stdio.h>
#define MAX 10

void bubble_sort(int arr[], int n)
{
    int i, j, temp;
    for (i = n - 1; i > 0; i--) {
        for (j = 0; j < i; j++) {
            if (arr[j] > arr[j + 1]) {
                temp = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = temp;
            }
        }
    }
}

int main()
{
    int arr[MAX] = { 2, 6, 4, 8, 10, 12, 89, 68, 45, 37 };
    bubble_sort(arr, MAX);
    for (int i = 0; i < MAX; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n");
    return 0;
}
```



3. 정렬

- 실습 3
  - 성적이 10개 입력되었을 때, 중앙값과 평균을 구해서 출력해보자.

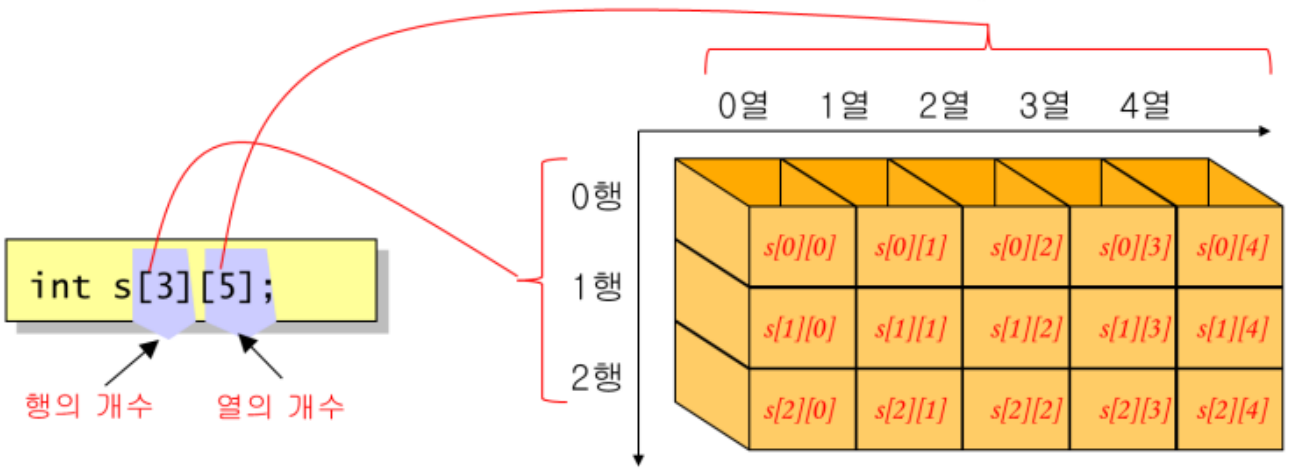
4. 탐색

- 배열 탐색
  - 내가 찾고자 하는 값이 어느 인덱스에 있는지?

```c
#include <stdio.h>
#define SIZE 10
int main(void)
{
    int key, i;
    int list[SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
    printf("탐색할 값을 입력하시오:");
    scanf("%d", &key);
    for (i = 0; i < SIZE; i++)
        if (list[i] == key)
            printf("탐색 성공 인덱스= %d\n", i);
    printf("탐색 종료\n");
    return 0;
}
```

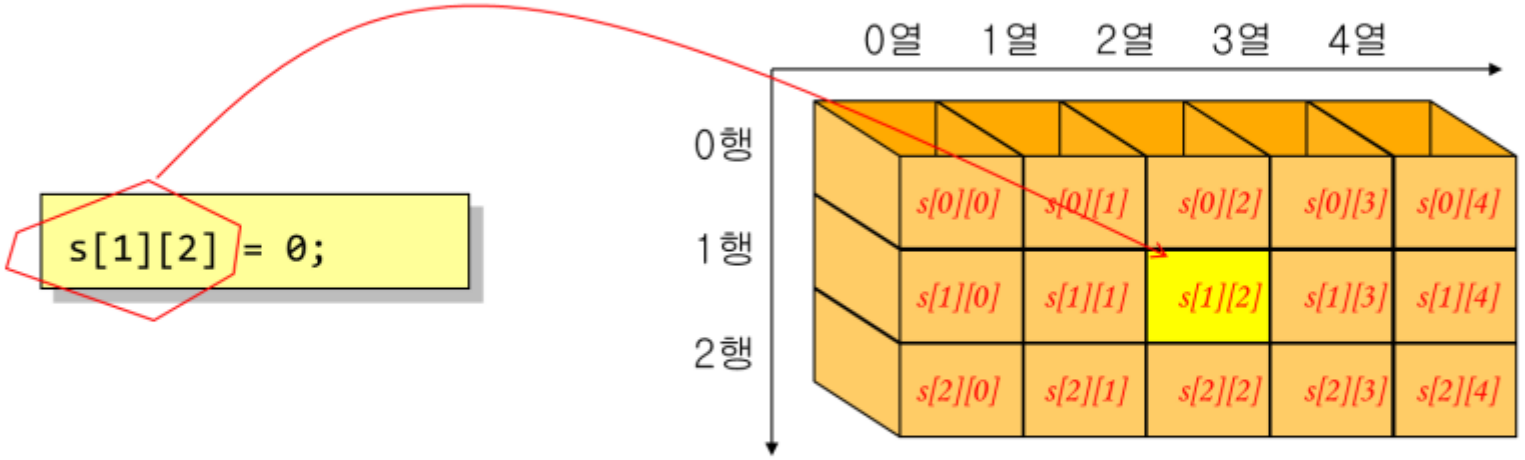
5. 다차원 배열

```
int s[10];      // 1차원 배열
int s[3][10];   // 2차원 배열
int s[5][3][10]; // 3차원 배열
```



5. 다차원 배열

❖ 요소 참조


s[1][2] = 0;
0열
1열
2열
3열
4열
0행
1행
2행
s[0][0]
s[0][1]
s[0][2]
s[0][3]
s[0][4]
s[1][0]
s[1][1]
s[1][2]
s[1][3]
s[1][4]
s[2][0]
s[2][1]
s[2][2]
s[2][3]
s[2][4]

5. 다차원 배열

❖ 초기화

```
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
};
```

```
int s[ ][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
};
```

```
int s[ ][5] = {
  0, 1, 2, 3, 4, 10, 11, 12, 13, 14, 20, 21, 22, 23, 24
};
```

5. 다차원 배열

- 실습4
  - 4*4 행렬 A와 B가 있을 때, 두 행렬의 곱을 연산하고 출력해보자
  - $$\begin{bmatrix} 1 & 2 & 5 & 6 \\ 3 & 4 & 7 & 8 \\ 9 & 1 & 4 & 5 \\ 2 & 3 & 6 & 7 \end{bmatrix} \times \begin{bmatrix} 5 & 7 & 5 & 6 \\ 3 & 4 & 6 & 8 \\ 9 & 3 & 2 & 5 \\ 2 & 3 & 6 & 7 \end{bmatrix} = \begin{bmatrix} 68 & 48 & 63 & 89 \\ 106 & 82 & 101 & 141 \\ 94 & 94 & 89 & 117 \\ 87 & 65 & 82 & 115 \end{bmatrix}$$

$$\begin{bmatrix} a_1 & a_2 & a_3 \\ b_1 & b_2 & b_3 \\ c_1 & c_2 & c_3 \end{bmatrix} \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ x_3 & y_3 & z_3 \end{bmatrix}$$

$$= \begin{bmatrix} a_1x_1 + a_2x_2 + a_3x_3 & a_1y_1 + a_2y_2 + a_3y_3 & a_1z_1 + a_2z_2 + a_3z_3 \\ b_1x_1 + b_2x_2 + b_3x_3 & b_1y_1 + b_2y_2 + b_3y_3 & b_1z_1 + b_2z_2 + b_3z_3 \\ c_1x_1 + c_2x_2 + c_3x_3 & c_1y_1 + c_2y_2 + c_3y_3 & c_1z_1 + c_2z_2 + c_3z_3 \end{bmatrix}$$

6. 배열 연습문제

- 실습 5
  - 사용자로부터 대문자 알파벳 'A'~'Z'가 20회 입력된다.
  - 사용자가 가장 많이 입력한 알파벳과 그 횟수를 출력해보자

6. 배열 연습문제

- 실습 6
  - 배열을 밀어내보자

  - 사용자가 정수 10개를 입력하여 이를 배열에 저장한다.
  - 사용자가 입력한 수 대로 배열을 오른쪽으로 밀어낸다.

  - int a[] = { 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 }

  - 1회 밀어냄
    - a[] == { 9, 0, 1, 2, 3, 4, 5, 6, 7, 8 }
  - 2회 밀어냄
    - a[] == { 8, 9, 0, 1, 2, 3, 4, 5, 6, 7 }