

# 프로그래밍연습 Lab 8

## 배열

---

[TA] 강성민, 김기현, 최석원, 최지은, 표지원

Department of Computer Science and Engineering

Seoul National University, Korea

2022/11/02

# 이번 장에서 학습할 내용

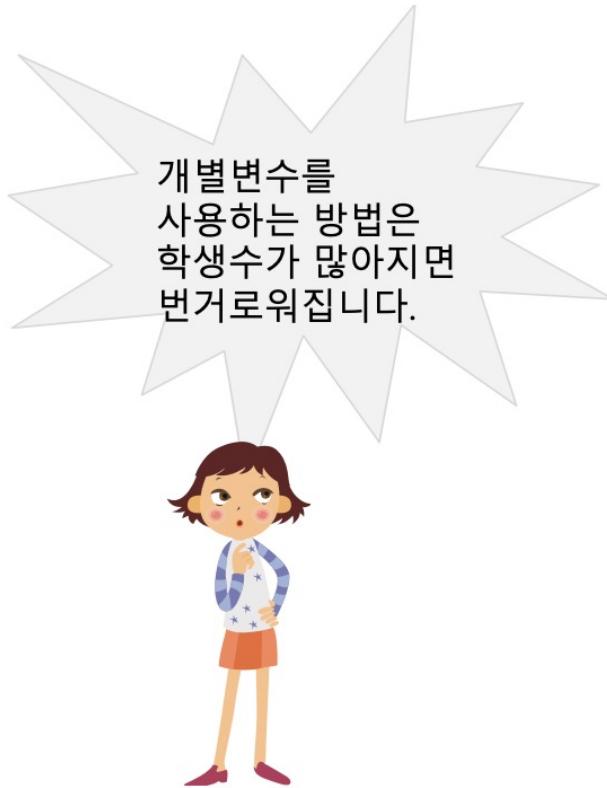
- 반복의 개념 이해
- 배열의 개념
- 배열의 선언과 초기화
- 일차원 배열
- 다차원 배열

배열을 사용하면 한 번에 여러 개의 값을 저장할 수 있는 공간을 할당받을 수 있다.



# 배열의 필요성

- 학생이 10명이 있고,
- 이들의 평균 성적을 계산한다고 가정하자.



방법 #1 : 개별 변수 사용

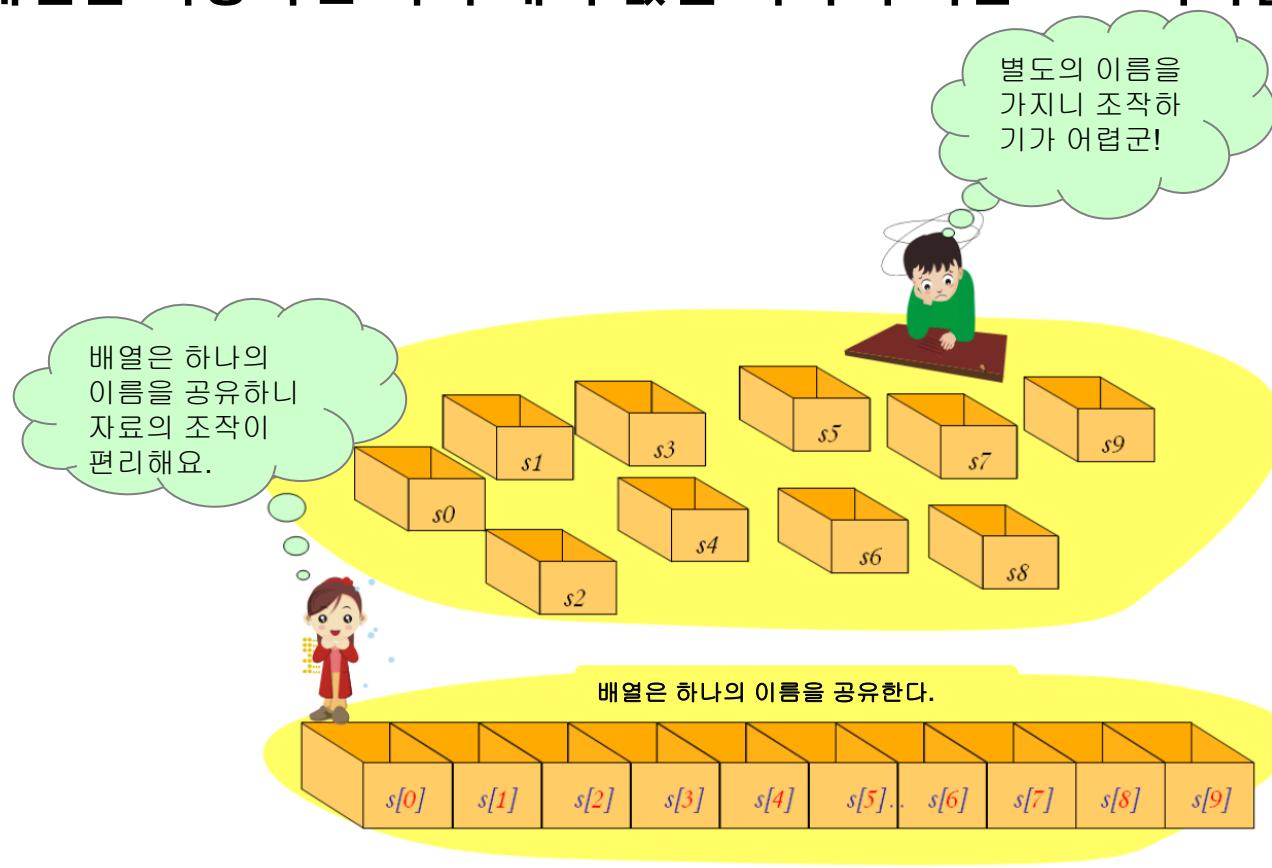
```
int s0;  
int s1;  
...  
int s9;
```

방법 #1 : 배열 사용

```
int[10];
```

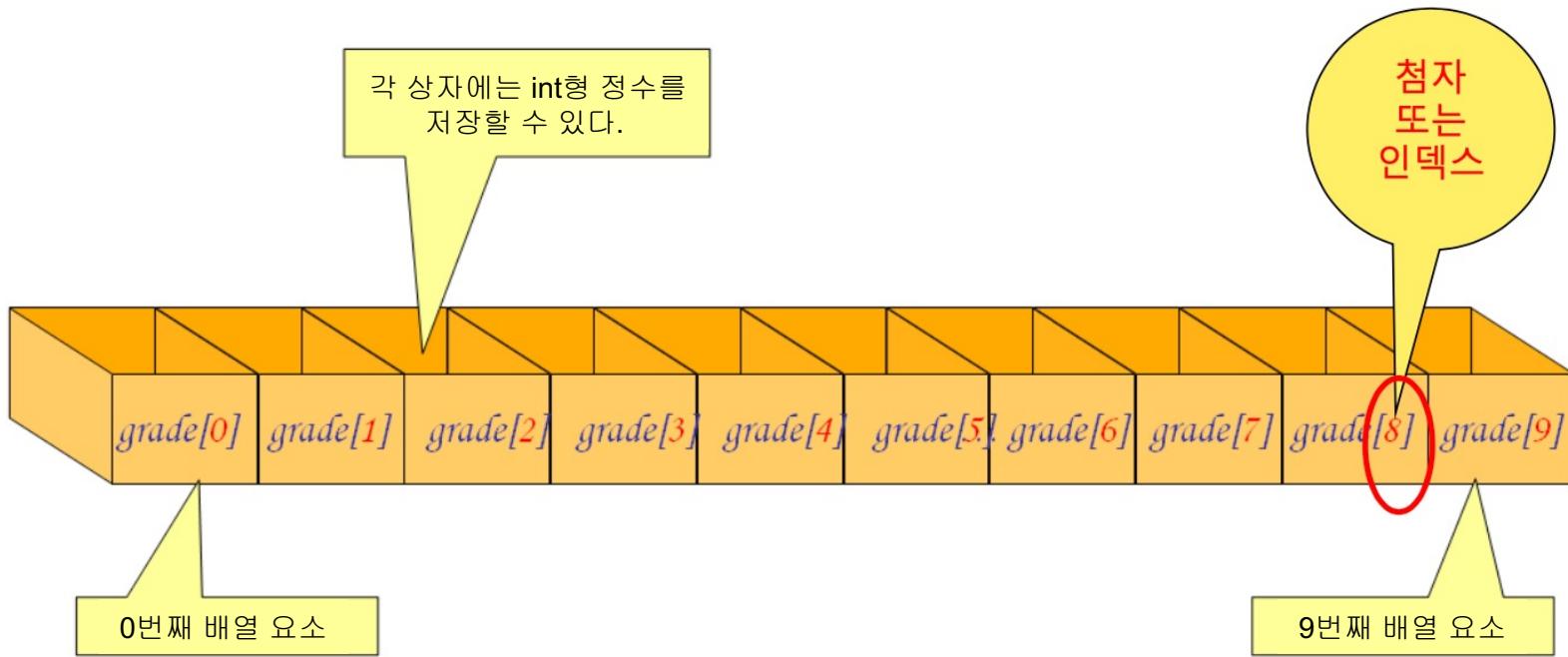
# 배열이란?

- **배열(array)** : 동일한 타입의 데이터가 여러 개 저장되어 있는 데이터 저장 장소
- 배열 안에 들어있는 데이터들은 정수로 되어 있는 번호(첨자)로 접근함
- 배열을 이용하면 여러 개의 값을 하나의 이름으로 처리할 수 있다.

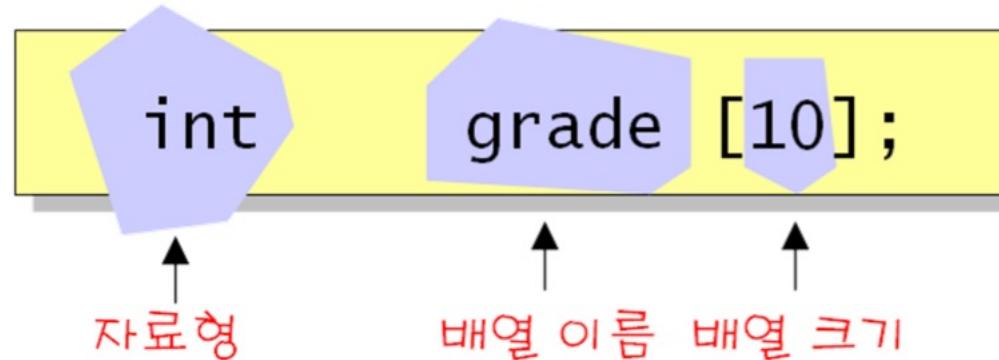


# 배열 원소와 인덱스

- **인덱스(index): 배열 원소의 번호**



# 배열의 선언



- 자료형: 배열 원소들이 int 형이라는 것을 의미
- 배열 이름: 배열을 사용할때 사용하는 이름이 grade
- 배열 크기: 배열 원소의 개수가 10개
- 인덱스(배열 번호)는 항상 0부터 시작한다.

# 배열 선언의 예

```
int grade[60];
```

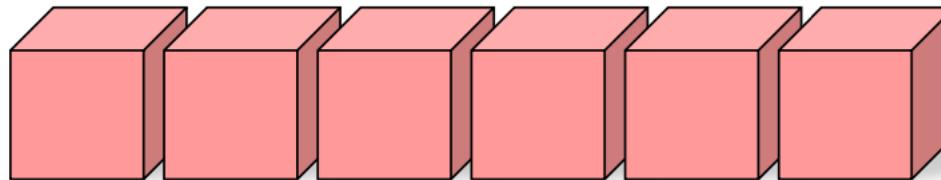
// 60개의 int형 값을 가지는 배열 grade

```
float cost[12];
```

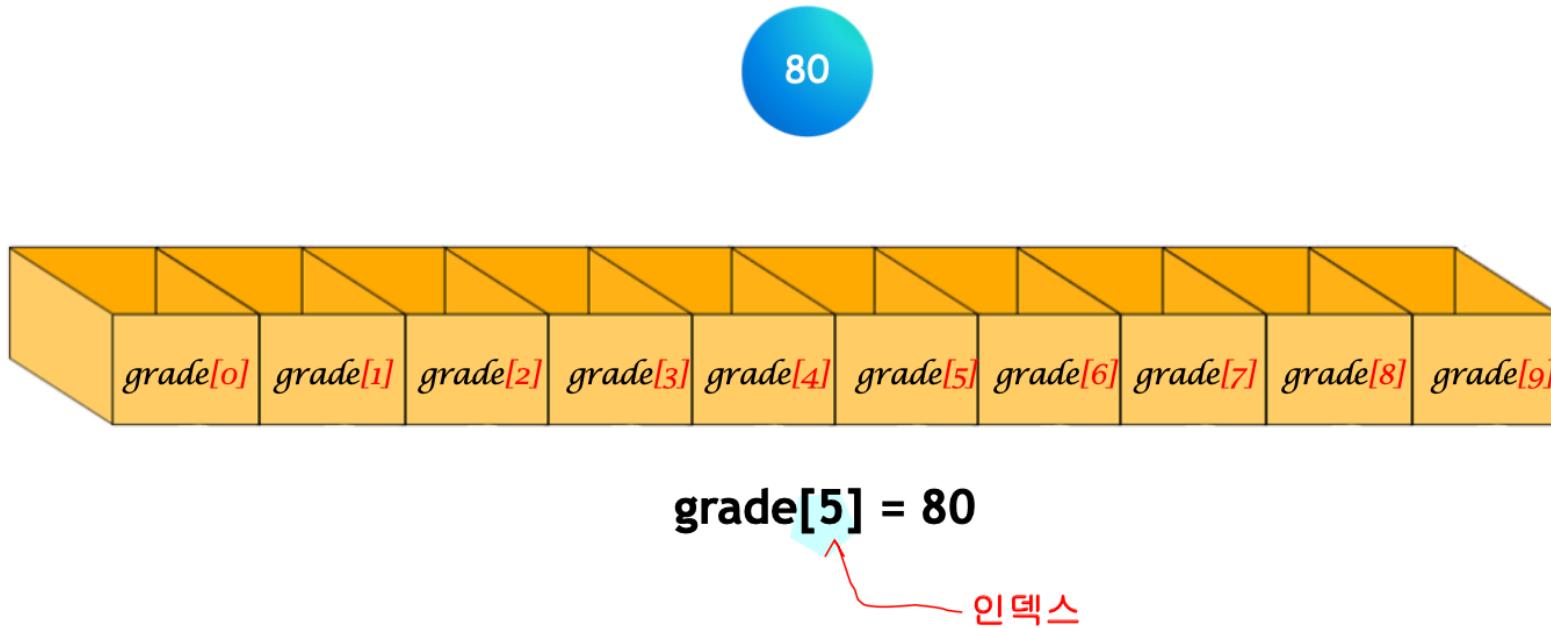
// 12개의 float형 값을 가지는 배열 cost

```
char name[50];
```

// 50개의 char형 값을 가지는 배열 name



# 배열 원소 접근



```
grade[5] = 80;  
grade[1] = grade[0];  
grade[i] = 100;      // i는 정수 변수  
grade[i+2] = 100;    // 수식이 인덱스가 된다.  
grade[index[3]] = 100; // index[]는 정수 배열
```

# 배열과 반복문

- 배열의 가장 큰 장점은 반복문을 사용해서 배열의 원소를 간편하게 처리할 수 있다는 점



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

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



# 배열 선언 예제(1) – 반복문을 이용한 원소 값 출력

```
#include <stdio.h>

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

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

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

    return 0;
}
```

- jechoi@labserver lab8 % ./array\_1  
grade[0] = 10  
grade[1] = 20  
grade[2] = 30  
grade[3] = 40  
grade[4] = 50

# 배열 선언 예제(2) – 배열 원소에 랜덤 값 저장

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

int main(void){
    int i;
    int grade[SIZE];

    for(i=0; i < SIZE; i++){
        grade[i] = rand() % 100;
    }

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

    return 0;
}
```

```
jechoi@labserver lab8 % ./array_2
grade[0] = 7
grade[1] = 49
grade[2] = 73
grade[3] = 58
grade[4] = 30
```

# 배열 선언 예제(3) – 배열 원소에 입력 값 저장

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

int main(void){
    int i;
    int grade[SIZE];
    printf("5명의 점수를 입력하시오 \n");

    for(i=0; i < SIZE; i++){
        scanf("%d", &grade[i]);
    }

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

    return 0;
}
```

- jechoi@labserver lab8 % ./array\_3  
5명의 점수를 입력하시오  
35  
45  
60  
80  
100  
grade[0] = 35  
grade[1] = 45  
grade[2] = 60  
grade[3] = 80  
grade[4] = 100

# 배열 선언 예제(4)

- 반복문을 이용하여 5명의 성적을 배열에 입력 받고, 성적 평균을 출력

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

int main(void){
    int i, average;
    int sum = 0;
    int grade[STUDENTS];

    for(i=0; i < STUDENTS; i++){
        printf("학생들의 성적을 입력하시오: ");
        scanf("%d", &grade[i]);
    }

    for(i=0; i < STUDENTS; i++){
        /* 배열의 원소 덧셈 코드 작성 */
    }

    /* 학생들의 성적 평균 계산 코드 작성 */
    printf("성적 평균 = %d \n", average);

    return 0;
}
```

# 배열 선언 예제(4)

- 반복문을 이용하여 5명의 성적을 배열에 입력 받고, 성적 평균을 출력

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

int main(void){
    int i, average;
    int sum = 0;
    int grade[STUDENTS];

    for(i=0; i < STUDENTS; i++){
        printf("학생들의 성적을 입력하시오: ");
        scanf("%d", &grade[i]);
    }

    for(i=0; i < STUDENTS; i++){
        sum += grade[i];
    }

    average = sum / STUDENTS;
    printf("성적 평균 = %d \n", average);

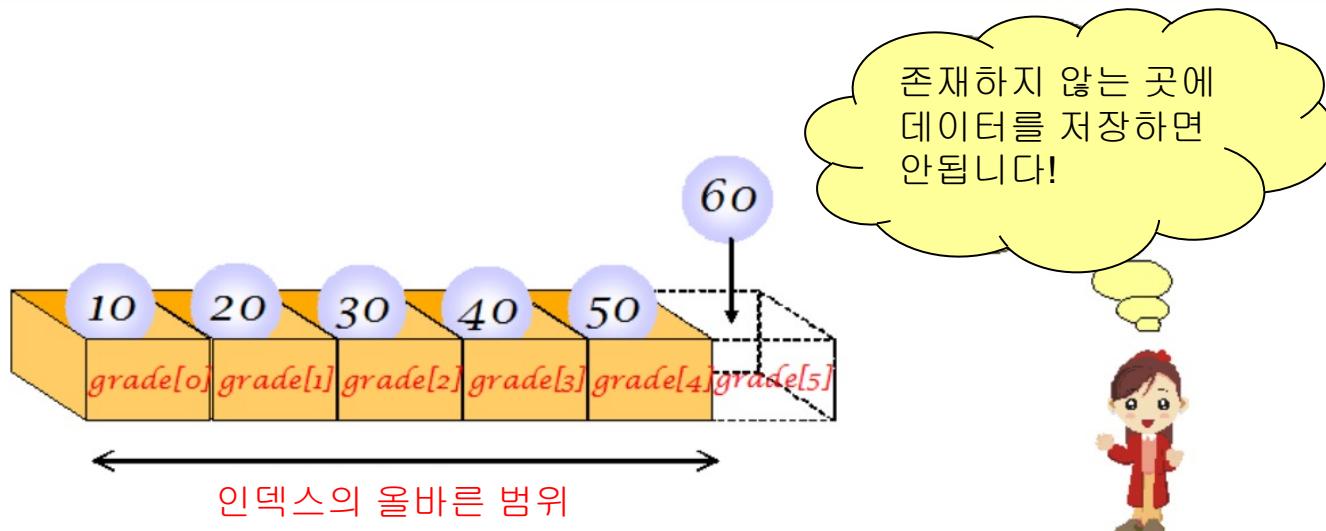
    return 0;
}
```

jechoi@labserver lab8 % ./array\_4  
학생들의 성적을 입력하시오 : 90  
학생들의 성적을 입력하시오 : 80  
학생들의 성적을 입력하시오 : 100  
학생들의 성적을 입력하시오 : 20  
학생들의 성적을 입력하시오 : 30  
성적 평균 = 64

# 잘못된 인덱스 문제

- 인덱스가 배열의 크기를 벗어나게 되면,  
프로그램에 치명적인 오류를 발생시킨다
- C에서는 프로그래머가 인덱스가 범위를 벗어나지 않았는지  
확인하고 책임을 져야 한다.

```
int grade[5];  
...  
grade[5] = 60; // 치명적인 오류!
```



# 잘못된 인덱스 예제

```
#include <stdio.h>

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

    grade[0] = 10;
    grade[1] = 20;
    grade[2] = 30;
    grade[3] = 40;
    grade[4] = 50;
    grade[5] = 60; // Error: Index 5 is past the end of the array

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

    return 0;
}
```

```
● jechoi@labserver lab8 % gcc -o wrong_index wrong_index.c
wrong_index.c:12:5: warning: array index 5 is past the end of the array (which contains 5 elements) [-Warray-bounds]
    grade[5] = 60;
    ^          ^
wrong_index.c:4:5: note: array 'grade' declared here
    int grade[5];
    ^
1 warning generated.
```

# 중간 점검

- 독립적인 여러 개의 변수 대신에 배열을 사용하는 이유는 무엇인가?
- N개의 원소를 가지는 배열의 경우, 첫번째 원소의 번호는 무엇인가?
- N개의 원소를 가지는 배열의 경우, 마지막 원소의 번호는 무엇인가?
- 배열 원소의 번호 혹은 위치를 무엇이라고 하는가?
- 배열의 크기보다 더 큰 인덱스를 사용하면 어떻게 되는가?
- 배열의 크기를 나타낼 때 변수를 사용할 수 있는가?

# 배열의 초기화

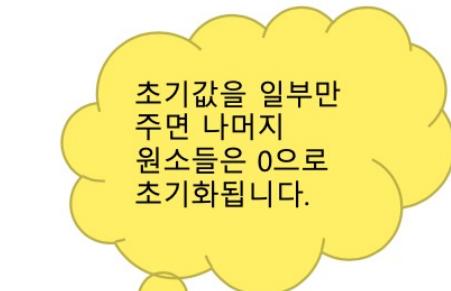
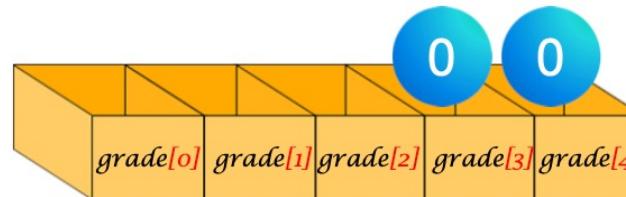
- $\text{int grade[5] = \{ 10,20,30,40,50 \};}$

$\text{int grade[5] = \{ }$   $\} ;$



- $\text{int grade[5] = \{ 10,20,30 \};}$

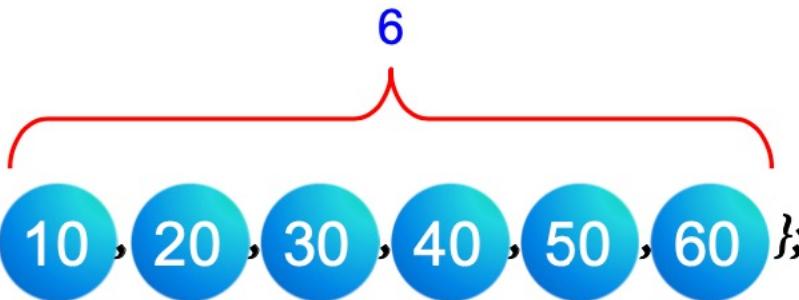
$\text{int grade[5] = \{ }$   $\} ;$



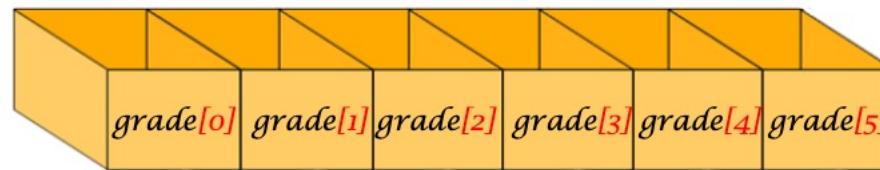
# 배열의 초기화

- 배열의 크기가 주어지지 않으면 자동적으로 초기값의 개수 만큼이 배열의 크기로 잡힌다.

`int grade[ ] = { 10, 20, 30, 40, 50, 60 };`



A diagram illustrating the initialization of an array. It shows a horizontal row of six blue circles, each containing a white numerical value: 10, 20, 30, 40, 50, and 60. Above this row, a red curly brace spans across all six elements. Above the brace, the number '6' is written in blue, indicating the size of the array. Below the row of circles, the code `int grade[ ] = { 10, 20, 30, 40, 50, 60 };` is displayed in black text.



# 배열의 초기화 예제

```
#include <stdio.h>

int main(void){
    int i;
    int grade_A[5] = {10, 20, 30, 40, 50};
    int grade_B[5];
    int grade_C[5];

    for(i=0; i<5; i++){ // 전체 원소 초기화한 경우
        printf("grade_A[%d] = %d \n", i, grade_A[i]);
    }
    printf("-----\n");

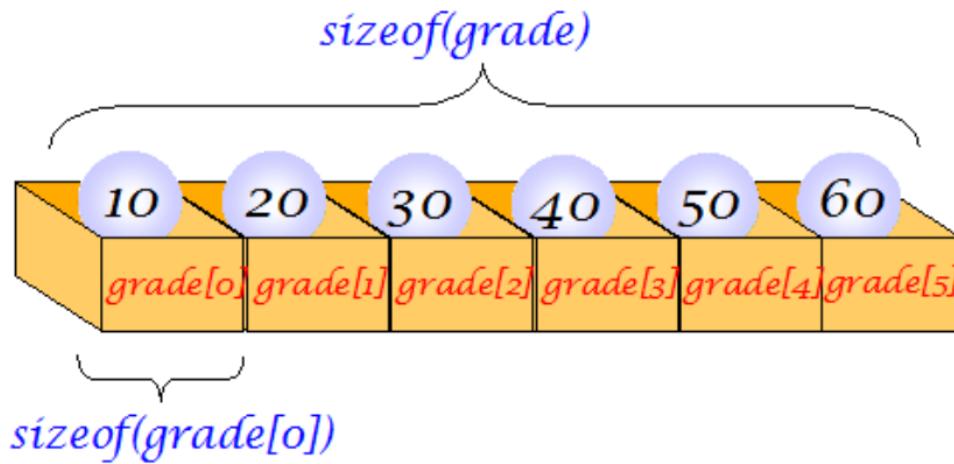
    for(i=0; i<5; i++){ // 일부 원소만 초기화한 경우
        printf("grade_B[%d] = %d \n", i, grade_B[i]);
    }
    printf("-----\n");

    for(i=0; i<5; i++){ // 초기화 안한 경우
        printf("grade_C[%d] = %d \n", i, grade_C[i]);
    }
}

return 0;
}
```

```
● jechoi@labserver lab8 % ./init_array
grade_A[0] = 10
grade_A[1] = 20
grade_A[2] = 30
grade_A[3] = 40
grade_A[4] = 50
-----
grade_B[0] = 15
grade_B[1] = 55
grade_B[2] = 0
grade_B[3] = 0
grade_B[4] = 0
-----
grade_C[0] = 377339904
grade_C[1] = 1
grade_C[2] = 377258000
grade_C[3] = 1
grade_C[4] = -1224878720
```

# 배열 원소의 개수 계산



```
int grade[] = { 1, 2, 3, 4, 5, 6 };
```

```
int i, size;
```

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

배열 원소 개수 자동 계산

```
for(i = 0; i < size ; i++)  
    printf("%d ", grade[i]);
```

# 배열의 복사



```
int grade[SIZE];  
int score[SIZE];  
score = grade;
```

// 컴파일 오류!

잘못된 방법



```
int grade[SIZE];  
int score[SIZE];  
int i;  
  
for(i = 0; i < SIZE; i++)  
    score[i] = grade[i];
```

원소를 일일이  
복사한다

올바른 방법

# 배열의 비교 예제

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

int main(void){
    int i;
    int a[SIZE] = {1, 2, 3, 4, 5};
    int b[SIZE] = {1, 2, 3, 4, 5};

    if(a == b){
        printf("잘못된 결과입니다. \n");
    }else{
        printf("잘못된 결과입니다. \n");
    }

    return 0;
}
```

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

int main(void){
    int i;
    int a[SIZE] = {1, 2, 3, 4, 5};
    int b[SIZE] = {1, 2, 3, 4, 5};

    for(i=0; i<SIZE; i++){
        if(a[i] != b[i]){
            printf("a[]와 b는 같지 않습니다. \n");
            return 0;
        }
    }
    printf("a[]와 b[]는 같습니다. \n");

    return 0;
}
```

- jechoi@labserver lab8 % gcc -o compare1 compare\_1.c  
compare\_1.c:9:10: warning: array comparison always evaluates to false [-Wtautological-compare]  
    if(a == b){  
        ^  
1 warning generated.
- jechoi@labserver lab8 % ./compare1  
잘못된 결과입니다 .

- jechoi@labserver lab8 % gcc -o compare2 compare\_2.c
- jechoi@labserver lab8 % ./compare2  
a[]와 b[]는 같습니다 .

# 중간 점검

- 배열  $a[6]$ 의 원소를 1, 2, 3, 4, 5, 6으로 초기화하는 문장을 작성하라
- 배열의 초기화에서 초기값의 개수가 배열 원소의 개수보다 적은 경우에는 어떻게 되는가? 또 반대로 많은 경우에는 어떻게 되는가?
- 배열이 크기를 주지 않고 초기값의 개수로 배열의 크기를 결정할 수 있는가?
- 배열  $a, b$ 를  $if(a == b)$ 와 같이 비교할 수 있는가?
- 배열  $a$ 에 배열  $b$ 를  $a=b$ 와 같이 대입할 수 있는가?

# 실습: 극장 예약 시스템

- 배열을 이용하여 간단한 극장 예약 시스템을 작성하기!
- 좌석은 10개
- 먼저 좌석 배치표를 보여준다.
- 예약이 끝난 좌석은 1로, 예약이 안된 좌석은 0으로 나타낸다.

<알고리즘>

**while(1)**

사용자로부터 예약 여부(y 또는 n)을 입력 받는다.

**if** 입력 == 'y'

현재의 좌석 배치표 **seats[]**를 출력한다.

좌석 번호 i를 사용자로부터 입력받는다.

**if** 좌석 번호가 올바르면

**seats[i] = 1**

**else**

에러메시지를 출력한다.

**else**

종료한다.

jechoi@labserver lab8 % ./box\_office  
좌석을 예약 하시겠습니까? (y 또는 n) y

-----  
1 2 3 4 5 6 7 8 9 10

-----  
0 0 0 0 0 0 0 0 0 0

몇 번째 좌석을 예약 하시겠습니까?  
예약되었습니다.

좌석을 예약 하시겠습니까? (y 또는 n) y

-----  
1 2 3 4 5 6 7 8 9 10

-----  
1 0 0 0 0 0 0 0 0 0

몇 번째 좌석을 예약 하시겠습니까?  
이미 예약된 자리입니다.

좌석을 예약 하시겠습니까? (y 또는 n) n

# 실습1: 극장 좌석 예약

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

int main(void){
    char check;      // 예약 진행 여부 확인 (y or n)
    int seat_num;    // 원하는 좌석 입력 (1~10)
    int i;
    int seats[SIZE] = {0};

    while(1){
        // 사용자로부터 예약 여부(y 또는 n)를 입력받는다.
        printf("좌석을 예약하시겠습니까? (y 또는 n) ");
        scanf(" %c", &check);

        if(check == 'y'){ /* 좌석을 예약하려는 경우 */
            /* 여기에 코드를 작성하시오 */
        }else if(check == 'n'){
            return 0; /* 좌석을 예약하지 않는 경우 프로그램 종료 */
        }
    }
    return 0;
}
```

if(check == 'y'){ /\* 좌석을 예약하려는 경우 \*/  
 /\* 여기에 코드를 작성하시오 \*/  
}else if(check == 'n'){

 return 0; /\* 좌석을 예약하지 않는 경우 프로그램 종료 \*/  
}



# 실습1: 극장 좌석 예약 (알고리즘)

```
if(check == 'y'){      /*좌석을 예약하려는 경우*/
    printf("----- \n");
    printf(" 1 2 3 4 5 6 7 8 9 10 \n");
    printf("----- \n");
    for(i=0; i<SIZE; i++){
        printf(" %d", seats[i]); // 현재의 좌석 배치표 seats[]를 출력한다.
    }
    printf("\n");
    printf("몇번째 좌석을 예약하시겠습니까?");
    scanf("%d", &seat_num); // 좌석 번호를 사용자로부터 입력받는다.

    if(seat_num <= 0 || seat_num > SIZE){
        printf("1부터 10사이의 숫자를 입력하세요. \n");
        continue;
    }

    if(seats[seat_num - 1] == 0){ // 현재 해당 좌석번호에 예약 가능하면
        seats[seat_num - 1] = 1; // seats[seat_num - 1] = 1;
        printf("예약되었습니다. \n");
    }else{                      // 예약이 불가능하면 안내 메시지 출력
        printf("이미 예약된 자리입니다. \n");
    }
}else if(check == 'n'){
    return 0; /* 좌석을 예약하지 않는 경우 프로그램 종료 */
}
```

# 도전 문제

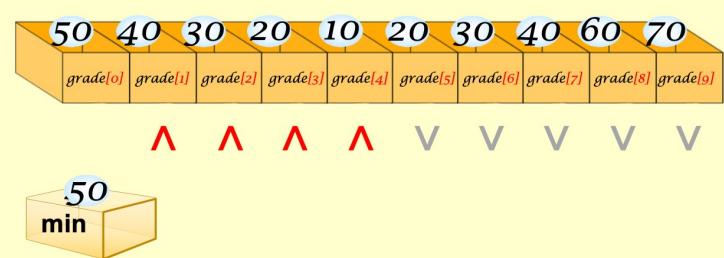
- 위의 프로그램에서는 한사람의 좌석만 예약이 가능하다.
- 한번에 2개의 좌석을 예약할 수 있도록 위의 프로그램을 변경하여 보자.

# 실습2: 최소값 찾기

- 우리는 인터넷에서 상품을 살 때,  
가격 비교 사이트를 통하여 가장 싼 곳을 검색한다.
- 일반적으로 배열에 들어 있는 정수 중에서 최소값을 찾는 문제와 같다.

## <알고리즘>

배열 `prices[]`의 원소를 난수로 초기화한다.  
일단 첫번째 원소를 최소값 `minimum` 이라 가정한다.  
`for(i=1; i<배열의 크기; i++)`.  
    `if (prices[i] < minimum )`  
        `minimum = prices[i]`  
반복이 종료되면 `minimum`에 최소값이 저장된다.



- jechoi@labserver lab8 % ./minvalue

```
-----  
1 2 3 4 5 6 7 8 9 10  
-----  
80 20 38 78 89 8 19 48 52 11
```

최 소 값 은 8입 니다 .

# 실습2: 최소값 찾기

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

int main(void){
    int prices[SIZE] = {0};
    int i, minimum;
    printf("-----\n");
    printf("1 2 3 4 5 6 7 8 9 10 \n");
    printf("-----\n");
    srand((unsigned)time(NULL));

    for(i=0; i<SIZE; i++){ /* 물건의 가격 출력 */
        prices[i] = (rand()%100)+1;
        printf("%-3d ", prices[i]);
    }
    printf("\n\n");

    minimum = prices[0]; // 첫번째 배열 원소를 최소값으로 가정
    for(i=1; i<SIZE; i++){
        if(prices[i] < minimum){ // 현재의 최소값보다 배열 원소가 작으면,
            minimum = prices[i]; // 배열 원소를 최소값으로 복사한다.
        }
    }

    printf("최소값은 %d입니다.\n", minimum);
    return 0;
}
```

# 도전 문제

- 위의 프로그램에서는 최소값을 계산하였다.
- 이번에는 배열의 원소 중에서 최대값을 찾도록 변경하여 보자!
- 변수 이름도 적절하게 변경하라

# 실습(3): 투표 집계하기

- 투표 결과를 컴퓨터를 이용해서 집계한다고 가정하자.
- 데이터의 빈도(frequency)를 계산하는 것과 동일
- 배열의 개념을 이용하면 손쉽게 구현할 수 있다.

## <알고리즘>

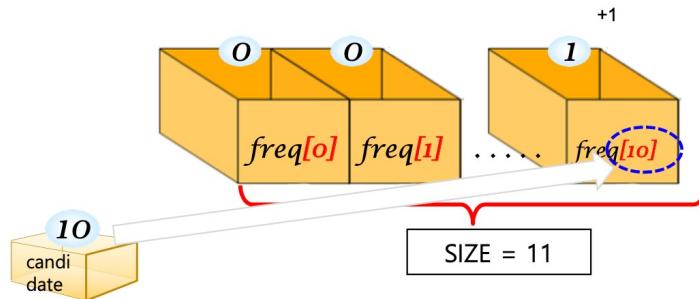
배열 freq[]의 원소를 0으로 초기화한다.

**while(1)**

    사용자로부터 후보자를 입력받는다.

    freq[candidate]++;

freq 배열의 내용을 출력한다.



● jechoi@labserver lab8 % ./vote

몇 번 후보자를 선택 하시겠습니까? (종료 -1):5

몇 번 후보자를 선택 하시겠습니까? (종료 -1):7

몇 번 후보자를 선택 하시겠습니까? (종료 -1):5

몇 번 후보자를 선택 하시겠습니까? (종료 -1):9

몇 번 후보자를 선택 하시겠습니까? (종료 -1):9

몇 번 후보자를 선택 하시겠습니까? (종료 -1):9

몇 번 후보자를 선택 하시겠습니까? (종료 -1):-1

후보자번호      득표 결과

1	0
2	0
3	0
4	0
5	2
6	0
7	1
8	0
9	3
10	0

# 실습(3): 투표 집계하기

```
#include <stdio.h>
#define SIZE 11

int main(void){
    int freq[SIZE] = {0};      // 빈도를 나타내는 배열
    int i, candidate;

    while(1){
        printf("몇번 후보자를 선택하시겠습니까? (종료-1):");
        scanf("%d", &candidate);
        if(candidate < 0) break;    // 음수이면 반복 종료
        freq[candidate]++;
    }

    printf("후보자번호  득표결과\n");
    for(i=1; i<SIZE; i++){
        printf("%5d %10d \n", i, freq[i]);
    }

    return 0;
}
```

# 도전문제

- c에서는 배열의 인덱스가 항상 0부터 시작한다.
- 하지만 일상 생활에서는 번호가 1부터 시작한다.
- 여기서 발생되는 문제가 상당히 있다.
- 위의 프로그램을 배열의 크기(SIZE)를 10으로 설정해서 변경해보자!

# ex) 배열 원소 역순 출력

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

int main(void){
    int data[SIZE];
    int i;

    for(i=0; i<SIZE; i++){ // 정수를 입력받는 루프
        printf("정수를 입력하시오:");
        scanf("%d", &data[i]);
    }

    for(i=SIZE-1; i >= 0; i--){ // 역순으로 출력하는 루프
        printf("%d\n", data[i]);
    }

    return 0;
}
```

- jechoi@labserver lab8 % ./reverse\_array  
정수를 입력하시오:10  
정수를 입력하시오:20  
정수를 입력하시오:30  
정수를 입력하시오:40  
정수를 입력하시오:50  
50  
40  
30  
20  
10

# ex) 막대 그래프

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

int main(void){
    int grade[STUDENTS] = {30, 20, 10, 40, 50};
    int i, s;

    for(i=0; i < STUDENTS; i++){
        printf("번호 %d: ", i);
        for(s=0; s < grade[i]; s++){
            printf("*");
        }
        printf("\n");
    }

    return 0;
}
```

jechoi@labserver lab8 % ./bar\_graph  
번호 0: \*\*\*\*\*  
번호 1: \*\*\*\*  
번호 2: \*\*  
번호 3: \*\*\*\*\*  
번호 4: \*\*\*\*\*

# ex) 빈도 계산

```
#include <stdio.h>
#define SIZE 101

int main(void){
    int freq[SIZE];
    int i, score;

    for(i=0; i<SIZE; i++){
        freq[i] = 0;
    }

    while(1){
        printf("숫자를 입력하시오(종료-1): ");
        scanf("%d", &score);
        if(score < 0) break;
        freq[score]++;
    }

    printf("  값  빈도 \n");

    for(i=0; i<SIZE; i++){
        printf("%3d %3d \n", i, freq[i]);
    }

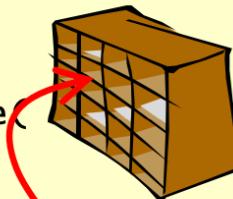
    return 0;
}
```

```
jechoi@labserver lab8 % ./freq
숫자를 입력하시오(종료-1): 0
숫자를 입력하시오(종료-1): 1
숫자를 입력하시오(종료-1): 99
숫자를 입력하시오(종료-1): 100
숫자를 입력하시오(종료-1): 100
숫자를 입력하시오(종료-1): -1
값 빈도
  0   1
  1   1
  2   0
  3   0
  4   0
  5   0
  6   0
  7   0
  8   0
  9   0
.
.
.
  95  0
  96  0
  97  0
  98  0
  99  1
 100  2
```

# 배열과 함수

- 배열의 경우에는 사본이 아닌 원본이 전달된다.

```
int main(void)
{
    ...
    get_average(    , int n);
    ...
}
```



배열 매개  
변수의 경우,  
원본이 직접  
참조됩니다.



```
int get_average(int score[], int n)
{
    ...
    sum += score[i];
    ...
}
```

# 배열과 함수

```
#include <stdio.h>
#define STUDENTS 5
int get_average(int score[], int n);

int main(void){
    int grade[STUDENTS] = {1, 2, 3, 4, 5};
    int avg;

    avg = get_average(grade, STUDENTS);
    printf("평균은 %d입니다.\n", avg);
    return 0;
}

int get_average(int score[], int n){
    int i;
    int sum=0;

    for(i=0; i<n; i++){
        sum += score[i];
    }
    return sum / n;
}
```

배열이 인수인 경우, 참조에 의한 호출

배열의 원본이 score[ ]로 전달

● jechoi@labserver lab8 % ./array\_func  
평균은 3입니다.

# 배열이 함수의 인수인 경우

```
1 #include <stdio.h>
2 #define SIZE 7
3
4 void square_array(int a[], int size);
5 void print_array(int a[], int size);
6
7 int main(void){
8     int list[SIZE] = {1, 2, 3, 4, 5, 6, 7};
9
10    print_array(list, SIZE);
11    square_array(list, SIZE);
12    print_array(list, SIZE);
13
14    return 0;
15 }
```

배열은 원본이 전달된다. (인수: 배열)

```
16
17    void square_array(int a[], int size){
18        int i;
19        for(i=0; i<size; i++){
20            a[i] = a[i] * a[i];
21        }
22    }
23
24    void print_array(int a[], int size){
25        int i;
26        for(i=0; i<size; i++){
27            printf("%3d ", a[i]);
28        }
29    }
30 }
```

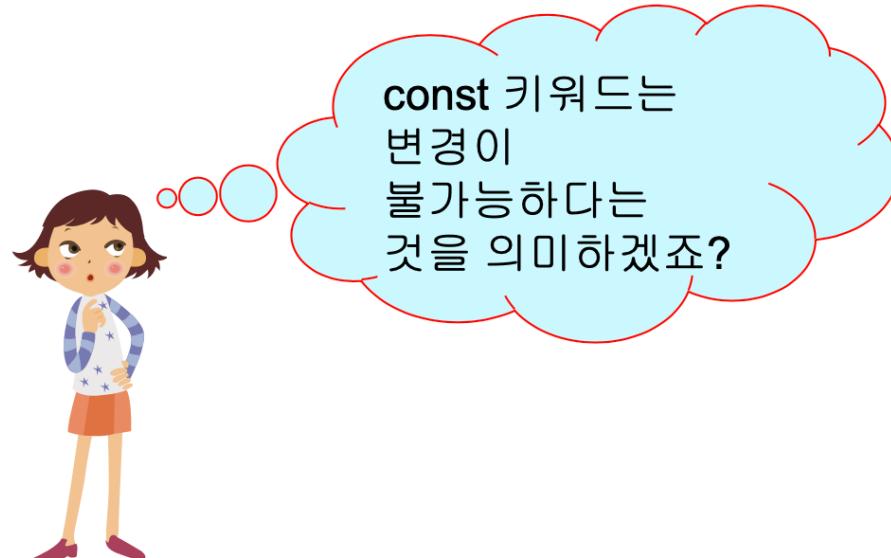
배열은 원본이 a[ ]로 전달됨

```
● jechoi@labserver lab8 % ./array_func2
      1   2   3   4   5   6   7
      1   4   9  16  25  36  49
```

# 원본 배열의 변경을 금지하는 방법

```
void print_array(const int a[], int size)
{
    ...
    a[0] = 100;           // 컴파일 오류!
}
```

함수 안에서 a[]는 변경할 수 없다.



const 키워드는  
변경이  
불가능하다는  
것을 의미하겠죠?

# 중간 점검

- 배열을 함수로 전달하면 원본이 전달되는가?  
아니면 복사본이 전달되는가?
- 함수가 전달받은 배열을 변경하지 못하게 하려면 어떻게 해야 하는가?

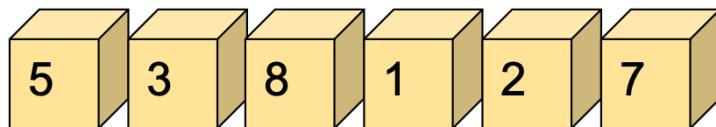
# 정렬이란?

- 정렬은 물건을 크기순으로 오름차순이나 내림차순으로 나열하는 것
- 정렬은 컴퓨터 공학분야에서 가장 기본적이고 중요한 알고리즘중의 하나
- 정렬은 자료 탐색에 있어서 필수적이다.  
(예) 만약 사전에서 단어들이 정렬이 안되어 있다면?



# 선택 정렬(selection sort) - ex) 오름차순

- 정렬이 안된 숫자들중에서 최소값을 선택하여 배열의 첫번째 요소와 교환



# 선택 정렬(selection sort) – 오름차순 예제

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

int main(void){
    int list[SIZE] = {3, 2, 9, 7, 1, 4, 8, 0, 6, 5}; // 원래의 배열
    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]); // 오름차순으로 정렬된 배열이 출력됨
    }

    printf("\n");
    return 0;
}
```

내부 for루프로서 (i+1)번째 원소부터 배열의 마지막 원소중에서 최소값을 찾는다.  
현재의 최소값과 비교하여 더 작은 정수가 발견되면 그 정수가 들어 있는 인덱스를 least에 저장한다.

list[i]와 list[least]를 서로 교환

● jechoi@labserver lab8 % ./selection\_sort  
0 1 2 3 4 5 6 7 8 9

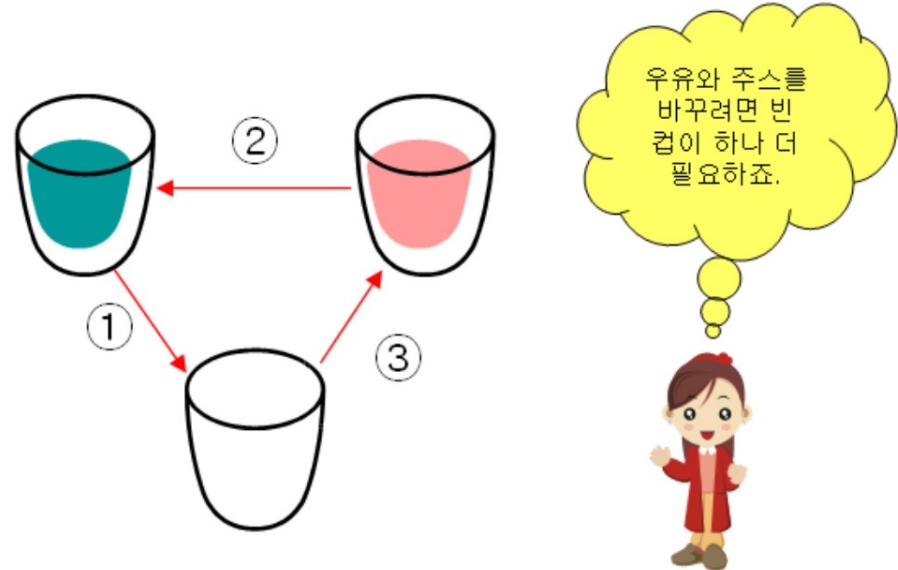
# 변수의 값을 서로 교환할 때

## ■ 다음과 같이 하면 안됨

- `grade[i] = grade[least];` // `grade[i]`의 기존값은 파괴된다!
- `grade[least] = grade[i];`

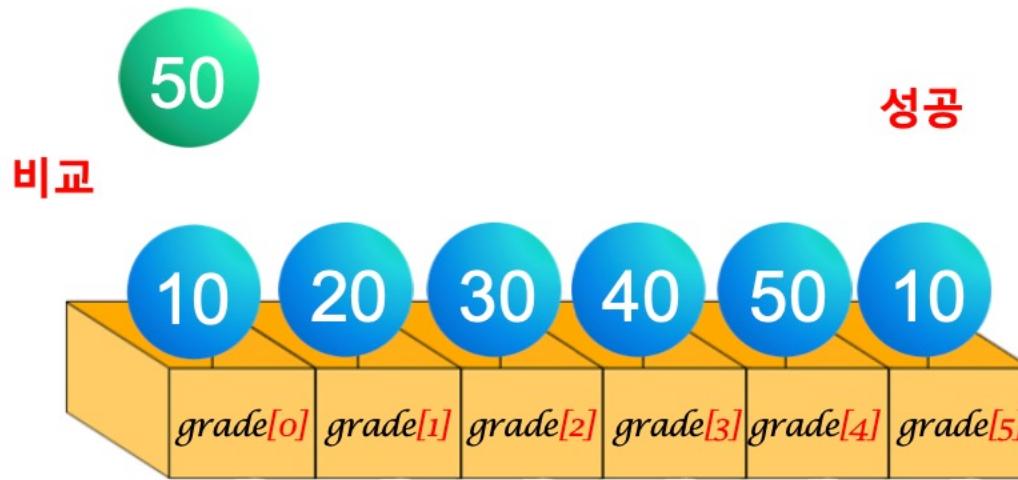
## ■ 올바른 방법

- `temp = list[i];`
- `list[i] = list[least];`
- `list[least] = temp;`



# 순차 탐색(sequential search)

- 배열의 원소를 순서대로 하나씩 꺼내서 탐색키와 비교하여 원하는 값을 찾아가는 방법



# 순차 탐색(sequential search) 예제

```
#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);
    }

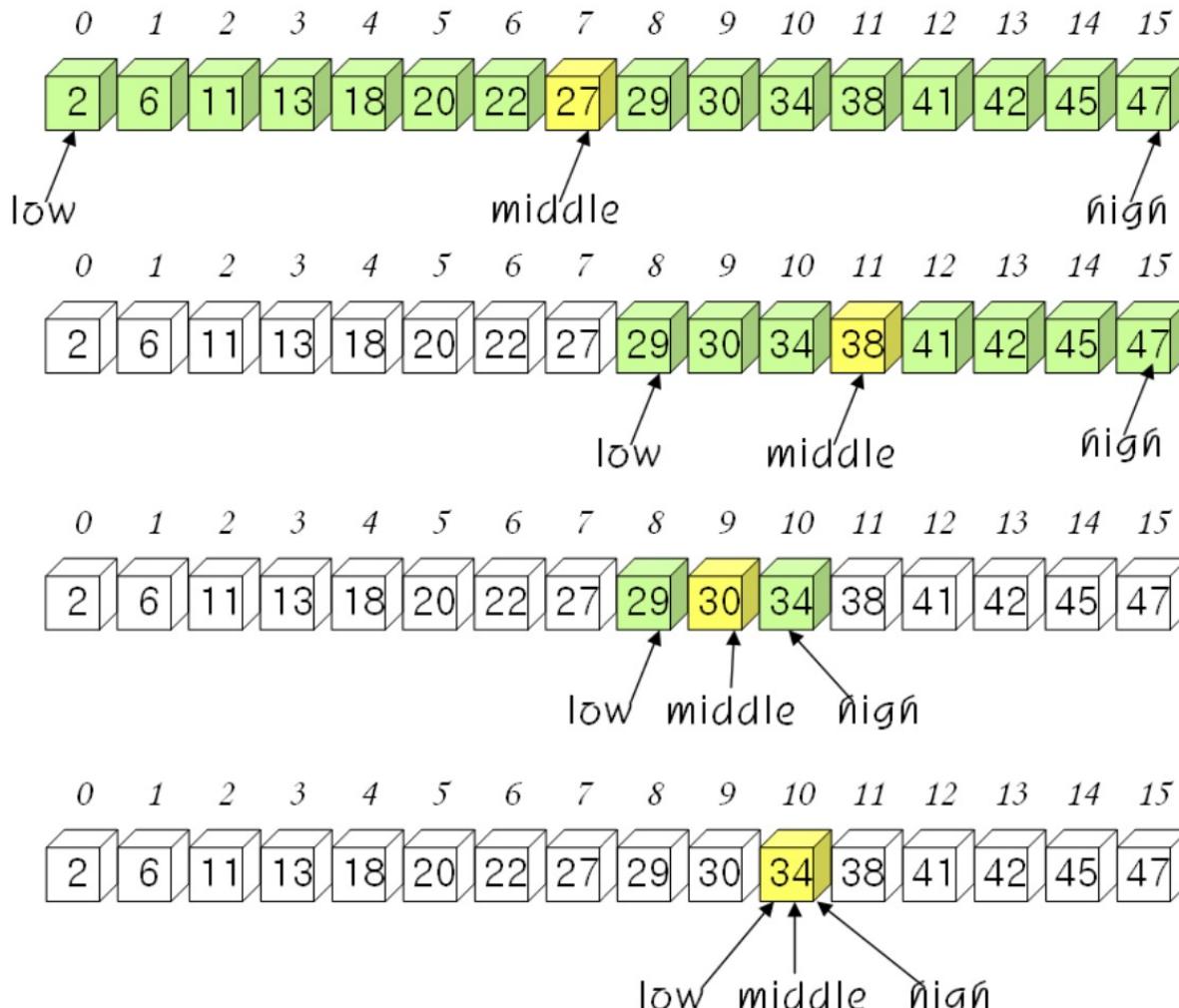
    printf("탐색 종료\n");
    return 0;
}
```

for루프를 이용하여 list[i]와 key를 비교하는 연산을 배열의 크기만큼 반복한다.  
만약 list[i]와 key가 같으면 탐색은 성공되고 키값이 발견된 배열의 인덱스를 출력한다.

jechoi@labserver lab8 % ./sequential  
탐색 할 값을 입력 하시오 :7  
탐색 성공 인덱스 = 6  
탐색 종료

# 이진 탐색(binary search)

- 정렬된 배열의 중앙에 위치한 원소와 비교 되풀이



# 이진 탐색(binary search) 예제

```
#include <stdio.h>
#define SIZE 16
int binary_search(int list[], int n, int key);

int main(void){
    int key;
    int grade[SIZE] = {2,6,11,13,18,20,22,27,29,30,34,38,41,42,45,47};
    printf("탐색할 값을 입력하시오:");
    scanf("%d", &key);
    printf("탐색 결과= %d\n", binary_search(grade, SIZE, key));

    return 0;
}

int binary_search(int list[], int n, int key){
    int low, high, middle;
    low = 0;
    high = n-1;

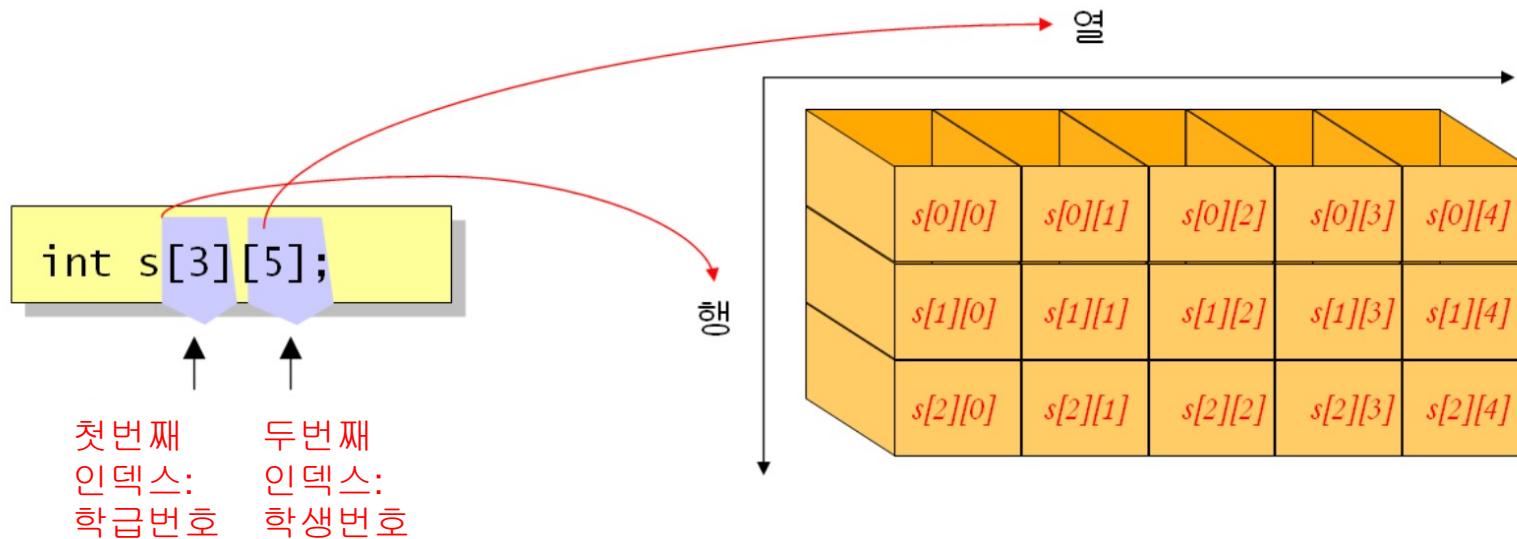
    while(low <= high){ // 아직 숫자들이 남아있으면
        printf("[%d %d]\n", low, high); // 하한과 상한을 출력한다.
        middle = (low+high)/2; // 중간 위치를 계산한다.

        if(key == list[middle]){ // 일치하면 탐색 성공
            return middle;
        }else if(key > list[middle]){
            low = middle + 1; // 새로운 값으로 low 설정
        }else{
            high = middle - 1; // 새로운 값으로 high 설정
        }
    }
    return -1; // 탐색이 실패했음을 의미
}
```

- jechoi@labserver lab8 % ./binary\_search  
탐색할 값을 입력 하 시 오 :34  
[0 15]  
[8 15]  
[8 10]  
[10 10]  
탐색 결과 = 10
- jechoi@labserver lab8 % ./binary\_search  
탐색할 값을 입력 하 시 오 :100  
[0 15]  
[8 15]  
[12 15]  
[14 15]  
[15 15]  
탐색 결과 = -1

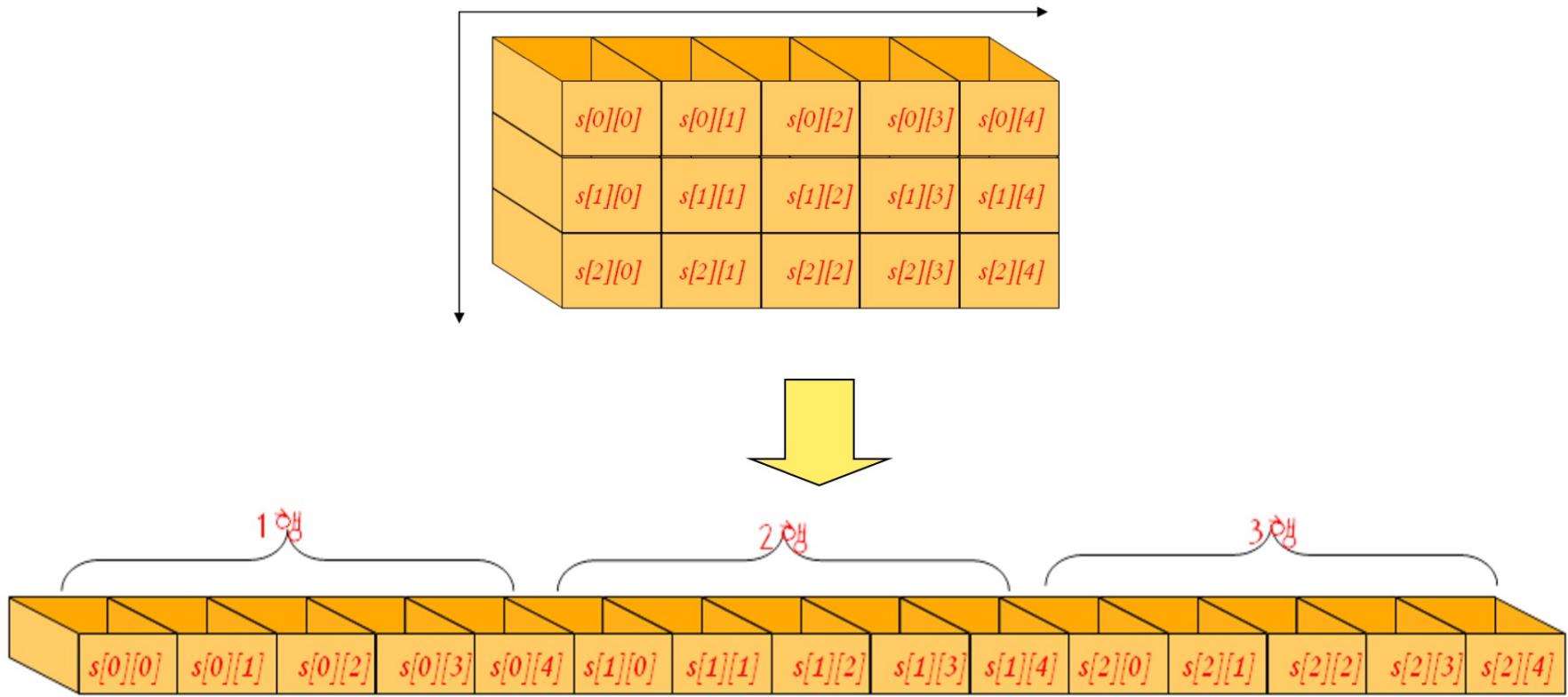
# 2차원 배열

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



# 2차원 배열의 구현

- 2차원 배열은 1차원적으로 구현된다.



# 2차원 배열의 활용

```
#include <stdio.h>

int main(void){
    int s[3][5];          // 2차원 배열 선언
    int i, j;             // 2개의 인덱스 변수
    int value = 0;         // 배열 원소에 저장되는 값

    for(i=0; i<3; i++){
        for(j=0; j<5; j++){
            s[i][j] = value++;
        }
    }

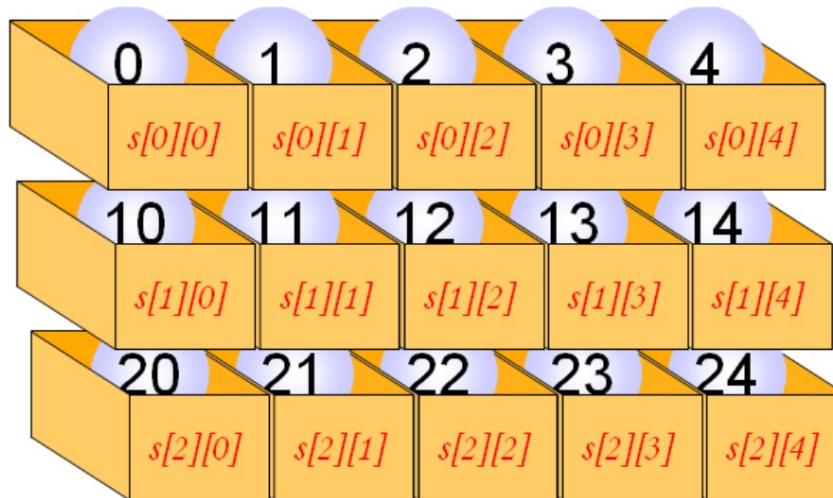
    for(i=0; i<3; i++){
        for(j=0;j<5;j++){
            printf("%d\n", s[i][j]);
        }
    }

    return 0;
}
```

```
● jechoi@labserver lab8 % ./2Darray
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
```

# 2차원 배열의 초기화(1)

```
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  
};
```



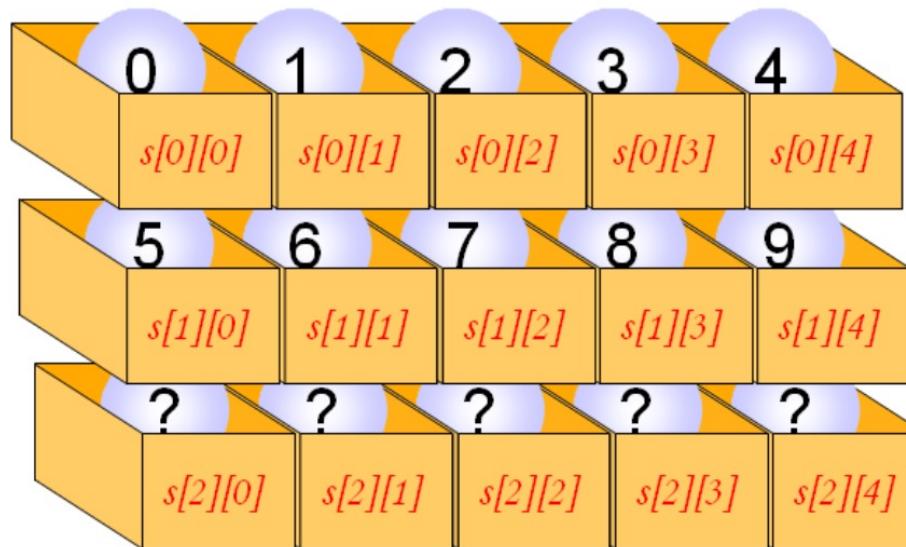
# 2차원 배열의 초기화(2)

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



# 2차원 배열의 초기화(3)

```
int s[ ][5] = {  
    0, 1, 2, 3, 4,           // 첫 번째 행의 원소들의 초기값  
    5, 6, 7, 8, 9,           // 두 번째 행의 원소들의 초기값  
};
```



# 3차원 배열

```
int s [6][3][5];
```



첫번째      두번째      세번째  
인덱스:      인덱스:      인덱스:  
학년번호      학급번호      학생번호

```
#include <stdio.h>

int main(void){
    int s[3][3][3];      // 3차원 배열선언
    int x, y, z;         // 3개의 인덱스 변수
    int i=1;              // 배열 원소에 저장되는 값

    for(z=0; z<3; z++){
        for(y=0; y<3; y++){
            for(x=0; x<3; x++){
                s[z][y][x] = i++;
                printf("s[%d][%d][%d] = %d\n", z, y, x, s[z][y][x]);
            }
        }
    }

    return 0;
}
```

```
jechoi@labserver lab8 % ./3D_array
s[0][0][0] = 1
s[0][0][1] = 2
s[0][0][2] = 3
s[0][1][0] = 4
s[0][1][1] = 5
s[0][1][2] = 6
s[0][2][0] = 7
s[0][2][1] = 8
s[0][2][2] = 9
s[1][0][0] = 10
s[1][0][1] = 11
s[1][0][2] = 12
s[1][1][0] = 13
s[1][1][1] = 14
s[1][1][2] = 15
s[1][2][0] = 16
s[1][2][1] = 17
s[1][2][2] = 18
s[2][0][0] = 19
s[2][0][1] = 20
s[2][0][2] = 21
s[2][1][0] = 22
s[2][1][1] = 23
s[2][1][2] = 24
s[2][2][0] = 25
s[2][2][1] = 26
s[2][2][2] = 27
```

# 다차원 배열 인수 예제 – 연간 매출의 합 계산

```
#include <stdio.h>
#define YEARS 3
#define PRODUCTS 5

int sum(int grade[][][PRODUCTS]);

int main(void){
    int sales[YEARS][PRODUCTS] = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} };
    int total_sale;
    total_sale = sum(sales);

    printf("총매출은 %d입니다.\n", total_sale);
    return 0;
}

int sum(int grade[][][PRODUCTS]){
    int y, p;
    int total = 0;

    for(y = 0; y < YEARS; y++){
        for(p = 0; p < PRODUCTS; p++){
            total += grade[y][p];
        }
    }
    return total;
}
```

jechoi@labserver lab8 % ./2D\_array2  
총매출은 45입니다.

# 다차원 배열 예제 – 전체 학생들의 평균 성적

```
#include <stdio.h>
#define CLASSES 3
#define STUDENTS 5

int main(void){
    int s[CLASSES][STUDENTS] = {
        { 0, 1, 2, 3, 4 }, //첫번째행의원소들의초기값
        { 10, 11, 12, 13, 14 }, // 두번째 행의 원소들의 초기값
        { 20, 21, 22, 23, 24 }, // 세번째 행의 원소들의 초기값
    };

    int clas, student, total, subtotal;
    total = 0;

    for(clas = 0; clas < CLASSES; clas++){
        subtotal = 0;
        for(student = 0; student < STUDENTS; student++){
            subtotal += s[clas][student];
        }
        printf("학급 %d의 평균 성적= %d\n", clas, subtotal / STUDENTS);
        total += subtotal;
    }
    printf("전체 학생들의 평균 성적= %d\n", total/(CLASSES * STUDENTS));
    return 0;
}
```

```
● jechoi@labserver lab8 % ./class_avg
학급 0의 평균 성적 = 2
학급 1의 평균 성적 = 12
학급 2의 평균 성적 = 22
전체 학생들의 평균 성적 = 12
```

# 행렬(Matrix)

- 행렬은 자연과학에서 많은 문제를 해결하는데 사용

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 8 & 9 & 1 \\ 7 & 0 & 5 \end{bmatrix}$$

$$B = \begin{bmatrix} 0 & 0 & 0 & 7 & 0 & 0 \\ 9 & 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 & 0 \end{bmatrix}$$

Mathematics - ELEMENTARY MATRIX OPERATIONS

OPERATIONS: MULTI. CONST ELEMENT IN 2<sup>nd</sup> Row By 7:

$A = \begin{bmatrix} 2 & 3 & 0 \\ 8 & 9 & 1 \\ 7 & 0 & 5 \end{bmatrix}$

1) FIND  $E \in 2 \times 2 \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix}$

2) PREMULT  $\begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix} \begin{bmatrix} 2 & 3 & 0 \\ 8 & 9 & 1 \\ 7 & 0 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 1(2)+0(8)+0(7) & 1(3)+0(9)+0(0) & 1(0)+0(1)+0(5) \\ \dots & \dots & \dots \end{bmatrix}$

# 다차원 배열을 이용한 행렬의 표현

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

int main(void){
    int A[ROWS][COLS] = { {2,3,0},
                          {8,9,1},
                          {7,0,5} };
    int B[ROWS][COLS] = { {1,0,0},
                          {1,0,0},
                          {1,0,0} };
    int C[ROWS][COLS];

    int r, c;

    for(r=0; r<ROWS; r++){ /* 두개의 행렬을 더한다. */
        for(c=0; c<COLS; c++){
            C[r][c] = A[r][c] + B[r][c];
        }
    }

    for(r=0; r<ROWS; r++){ /* 행렬C를 출력한다. */
        for(c=0; c<COLS; c++){
            printf("%d ", C[r][c]);
        }
        printf("\n");
    }
    return 0;
}
```

중첩 for 루프를 이용하여 행렬 A의 각 원소들과 행렬 B의 각 원소들을 서로 더하여 행렬 C에 대입한다.



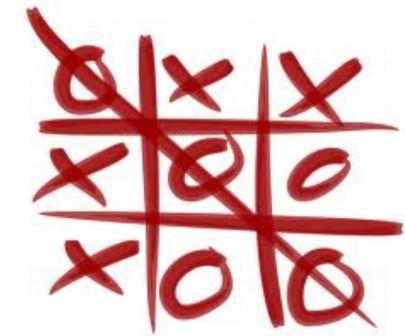
- jechoi@labserver lab8 % ./matrix1  
3 3 0  
9 9 1  
8 0 5

# 중간 점검

- 다차원 배열 `int a[3][2][10]`에는 몇개의 원소가 존재하는가?
- 다차원 배열 `int a[3][2][10]`의 모든 요소를 0으로 초기화하는 문장을 작성하시오.

# 실습: tic-tac-toe 게임

- 2명의 경기자가 오른쪽과 같은 보드를 이용해서 번갈아가며 O와 X를 놓는 게임.
- 같은 글자가 가로, 세로, 혹은 대각선 상에 놓이면 이긴다.



## <알고리즘>

보드를 초기화한다.

while(1)

    보드를 화면에 출력한다.  
    사용자로부터 좌표 x, y를 받는다.

    if(board[x][y]가 비어 있으면)

        if(현재 경기자가 'X'이면)  
            board[x][y] = 'X'

        else

            board[x][y] = 'Y'

    else

        오류메시지를 출력한다.

(x, y) 좌표 (종료 -1, -1): 1 1

---	---	---
X		
---	---	---
	0	
---	---	---

(x, y) 좌표 (종료 -1, -1): 2 2

---	---	---
X		
---	---	---
	0	
---	---	---
		X
---	---	---

(x, y) 좌표 (종료 -1, -1): -1 -1

# 실습: tic-tac-toe 게임 – main 함수

```
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 void init_board(char board[][3]);
5 int get_player_move(int palyer, char board[][3]);
6 void disp_board(char board[][3]);
7
8 int main(void){
9     char board[3][3];
10    int quit=0;
11    init_board(board);
12
13    do{
14        disp_board(board);
15        quit = get_player_move(0, board);
16        disp_board(board);
17        quit = get_player_move(1, board);
18    }while(quit == 0);
19
20    return 0;
21 }
```

# 실습: tic-tac-toe 게임- 보드 초기화/출력 함수

```
23 void init_board(char board[][][3]){
24     int x,y;
25     for(x=0; x<3; x++){
26         for(y=0; y<3; y++){
27             board[x][y] = ' ';
28         }
29     }
30 }
31
32 void disp_board(char board[3][3]){
33     int i;
34     for(i=0; i<3; i++){
35         printf(" ---|---|---\n");
36         printf(" %c | %c | %c \n",board[i][0], board[i][1], board [i][2]);
37     }
38     printf(" ---|---|---\n");
39 }
```

# 실습: tic-tac-toe 게임- player\_move 함수

```
40
41     int get_player_move(int player, char board[3][3]){
42         int x, y, done = 0;
43
44         while(done != 1){
45             printf("(x, y) 좌표(종료-1, -1): ");
46             scanf("%d %d", &x, &y);
47
48             if( x == -1 && y == -1 ){
49                 return 1;
50             }
51             if(board[x][y]== ' '){
52                 break;
53             }else{
54                 printf("잘못된 위치입니다.\n");
55             }
56         }
57         if( player == 0 ){
58             board[x][y] = 'X';
59         }else{
60             board[x][y] = 'O';
61         }
62
63         return 0;
64     }
```

# 도전 문제

- 보드를 분석해서 게임이 종료되었는지를 검사하는 함수를 추가하라
- 컴퓨터가 다음 수를 결정하도록 프로그램을 변경하라
  - 가장 간단한 알고리즘을 사용하면 된다.
  - ex) 비어있는 첫 번째 좌표에 놓는다.

# Q & A

