

개정3판

Visual
Studio
2017

쉽게 풀어쓴

C언어 EXPRESS

천인국 지음

제10장 배열



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.





배열

- 배열을 사용하면 한 번에 여러 개의 변수를 생성할 수 있다.
- `int s[10];`



변수

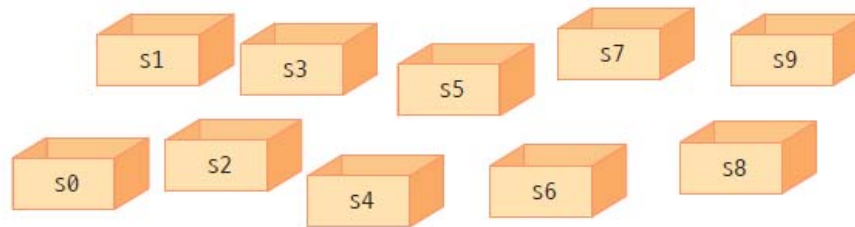


아파트



배열의 필요성

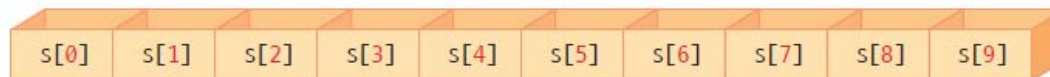
```
// 일반 변수 사용  
int s0;  
int s1;  
...  
int s9;
```



별도의 이름을 가지니
조작하기가 어렵군!



```
// 배열 사용  
int s[10];
```





배열 선언

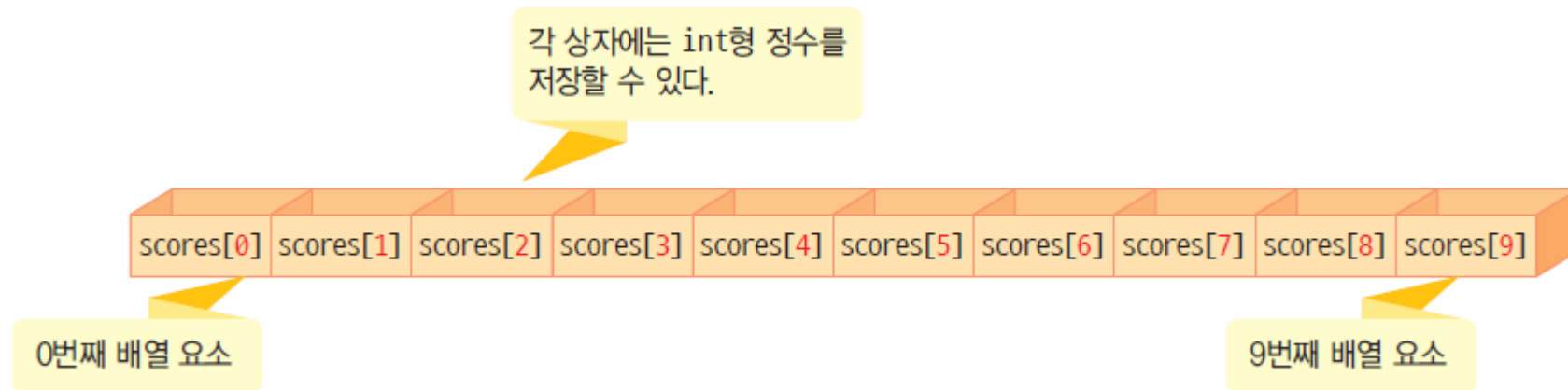
Syntax: 배열 선언





배열 원소와 인덱스

- **인덱스(index):** 배열 원소의 번호





배열 선언의 예

```
int score[60];
```

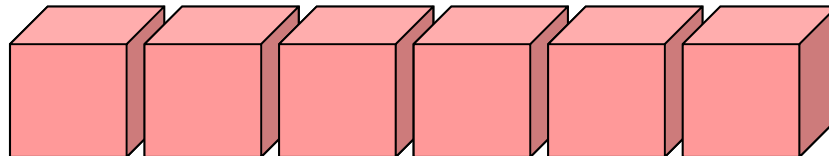
// 60개의 int형 값을 가지는 배열 score

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

// 12개의 float형 값을 가지는 배열 cost

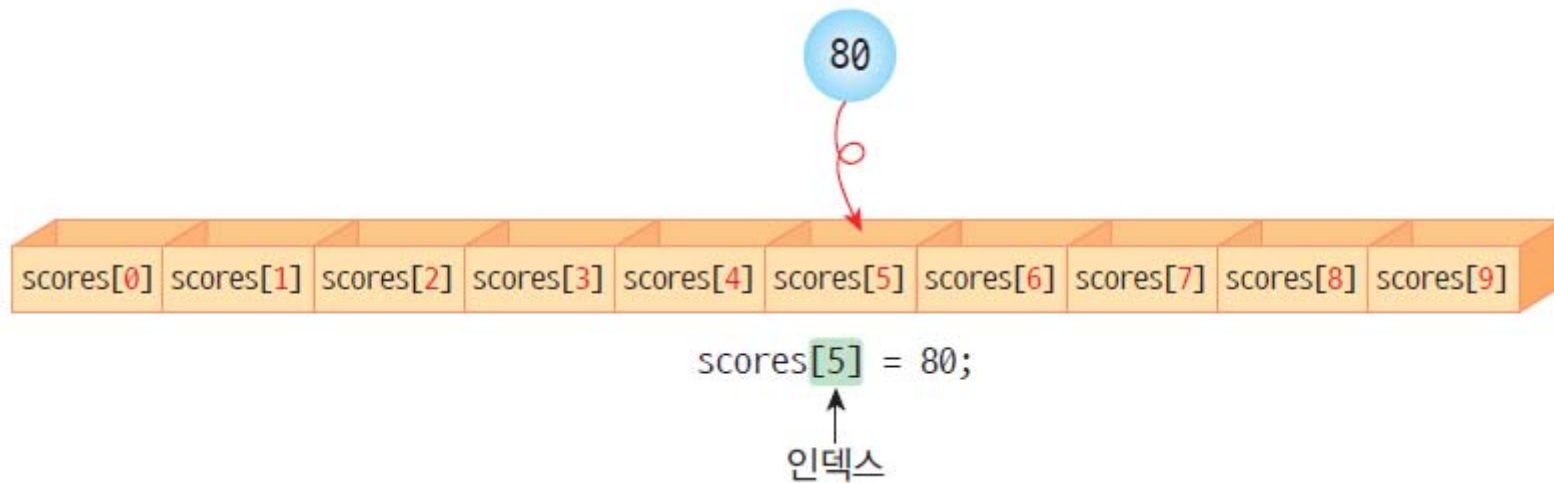
```
char name[50];
```

// 50개의 char형 값을 가지는 배열 name





배열 요소 접근



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



배열 선언 예제

```
#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, score[i]);
```

```
    return 0;
```

```
}
```

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



배열과 반복문

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



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

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





배열 선언 예제

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

```
int main(void)
{
```

```
    int i;
    int scores[SIZE];
```

```
    for(i = 0; i < SIZE; i++)
        scores[i] = rand() % 100;
```

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

```
    return 0;
```

```
}
```

```
scores[0]=41
scores[1]=67
scores[2]=34
scores[3]=0
scores[4]=69
```



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

```
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;
}
```

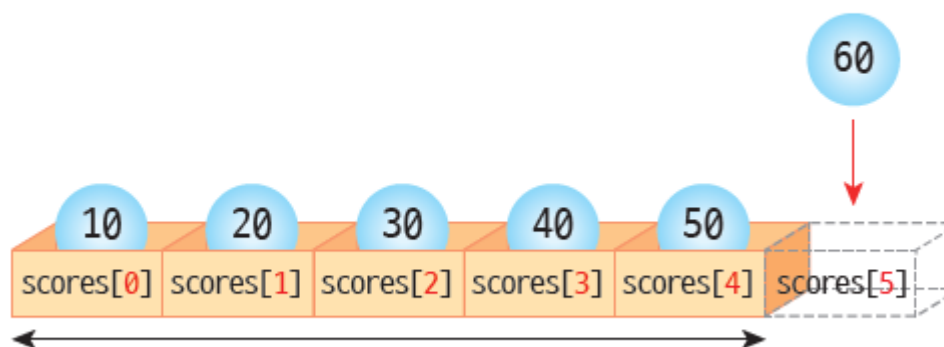
학생들의 성적을 입력하시오: 10
학생들의 성적을 입력하시오: 20
학생들의 성적을 입력하시오: 30
학생들의 성적을 입력하시오: 40
학생들의 성적을 입력하시오: 50
성적 평균 = 30



잘못된 인덱스 문제

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

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



존재하지 않는 곳에 데이터를 저장하면 안됩니다.





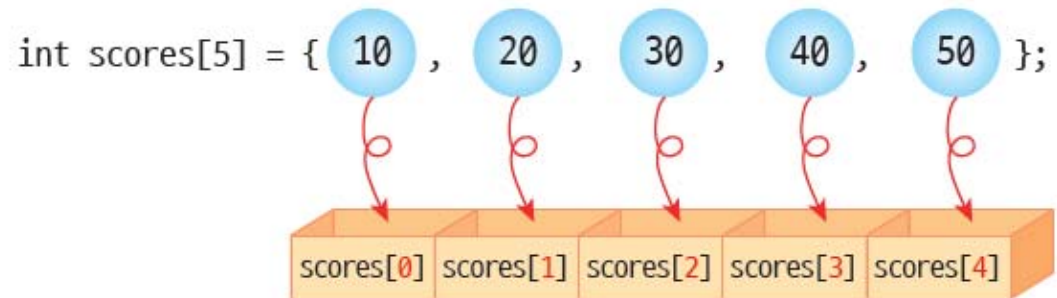
정간 점검

- 독립적인 여러 개의 변수 대신에 배열을 사용하는 이유는 무엇인가?
- n 개의 원소를 가지는 배열의 경우, 첫 번째 원소의 번호는 무엇인가?
- n 개의 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



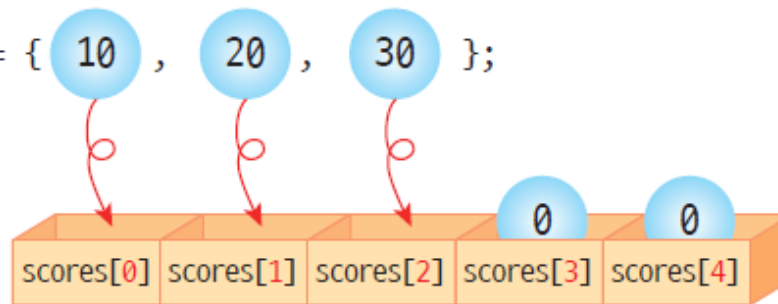
원소들의 초기값을 콤마로 분리하여 중괄호 안에 나열합니다.





배열의 초기화

```
int scores[5] = { 10 , 20 , 30 };
```

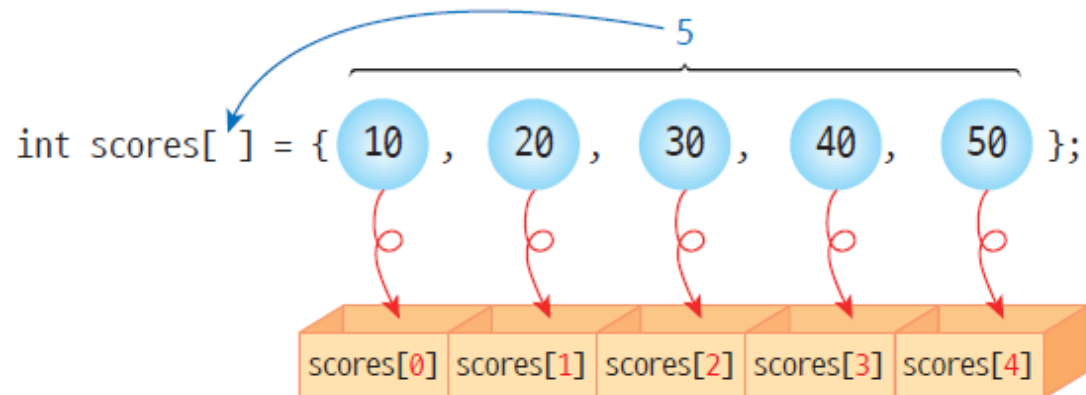


초기값을 일부만 주면 나머지
원소들은 0으로 초기화됩니다.





배열의 초기화



배열의 크기가 주어지지 않은 경우에는 초기값의 개수가 배열의 크기가 됩니다.





배열 초기화 예제

```
#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;
}
```

```
scores[0] = 31
scores[1] = 63
scores[2] = 62
scores[3] = 87
scores[4] = 14
```



배열 초기화 예제

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

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

    return 0;
}
```

```
scores[0] = 31
scores[1] = 63
scores[2] = 0
scores[3] = 0
scores[4] = 0
```



배열 초기화 예제

```
#include <stdio.h>
int main(void)
{
    int scores[5] ;
    int i;

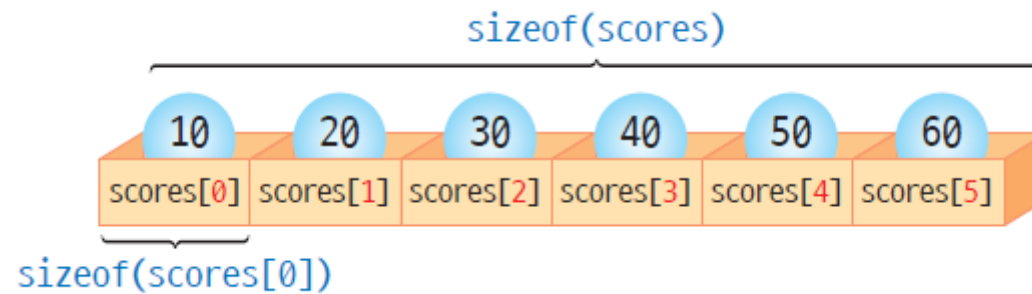
    for(i = 0; i < 5; i++)
        printf("scores[%d] = %d\n", i, scores[i]);

    return 0;
}
```

```
scores[0]=4206620
scores[1]=0
scores[2]=4206636
scores[3]=2018779649
scores[4]=1
```



배열 원소의 개수 계산



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

```
int i, size;
```

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

배열 원소 개수 자동 계산

```
for(i = 0; i < size ; i++)
```

```
    printf("%d ", scores[i]);
```



배열의 복사



```
int score[SIZE];
```

```
int score[SIZE];
```

```
score = score;
```

// 컴파일 오류!

잘못된 방법



```
int score[SIZE];
```

```
int score[SIZE];
```

```
int i;
```

```
for(i = 0; i < SIZE; i++)
```

```
    score[i] = score[i];
```

원소를 일일이
복사한다

올바른 방법



배열의 비교

```
#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 )                // ① 올바르게 않은 배열 비교
```

```
        printf("잘못된 결과입니다.\n");
```

```
    else
```

```
        printf("잘못된 결과입니다.\n");
```





배열의 비교



```
for(i = 0; i < SIZE ; i++) // ② 올바른 배열 비교
{
    if ( a[i] != b[i] )
    {
        printf("a[]와 b[]는 같지 않습니다.\n");
        return 0;
    }
}
printf("a[]와 b[]는 같습니다.\n");
return 0;
}
```

원소를 하나씩 비교한다



정가 점검

- 배열 `a[6]`의 원소를 1, 2, 3, 4, 5, 6으로 초기화하는 문장을 작성하라.
- 배열의 초기화에서 초기값이 개수가 배열 원소의 개수보다 적은 경우에는 어떻게 되는가? 또 반대로 많은 경우에는 어떻게 되는가?
- 배열의 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?





lab: 주사위 던지기

- 이 실습에서는 주사위를 1000번 던져서 각 면이 나오는 횟수를 출력하여 보자.





```
#include <stdio.h>
#include <stdlib.h>

#define SIZE 6

int main(void)
{
    int freq[SIZE] = { 0 };           // 주사위의 면의 빈도를 0으로 한다.
    int i;

    for(i = 0; i < 10000; i++)        // 주사위를 10000번 던진다.
        ++freq[ rand() % 6 ];        // 해당면의 빈도를 하나 증가한다.

    printf("=====\n");
    printf("면   빈도\n");
    printf("=====\n");

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

    return 0;
}
```



lab : 극장 예약 시스템

- 배열을 이용하여 간단한 극장 예약 시스템을 작성
- 좌석은 10개
- 예약이 끝난 좌석은 1로, 예약이 안 된 좌석은 0으로 나타낸다.





실행 결과

좌석을 예약하시겠습니까?(y 또는 n) y

1 2 3 4 5 6 7 8 9 10

0 0 0 0 0 0 0 0 0 0

몇번째 좌석을 예약하시겠습니까?1

예약되었습니다.

좌석을 예약하시겠습니까?(y 또는 n) y

1 2 3 4 5 6 7 8 9 10

1 0 0 0 0 0 0 0 0 0

몇번째 좌석을 예약하시겠습니까?1

이미 예약된 자리입니다. 다른 좌석을 선택하세요

좌석을 예약하시겠습니까?(y 또는 n) n



알고리즘

```
while(1)
```

사용자로부터 예약 여부(y 또는 n)를 입력받는다.

```
if 입력 == 'y'
```

현재의 좌석 배치표 `seats[]`를 출력한다.

좌석 번호 `i`를 사용자로부터 입력받는다.

```
if 좌석번호가 올바르면
```

```
seats[i]=1
```

```
else
```

에러 메시지를 출력한다.

```
else
```

종료한다.



실습: 극장 좌석 예약

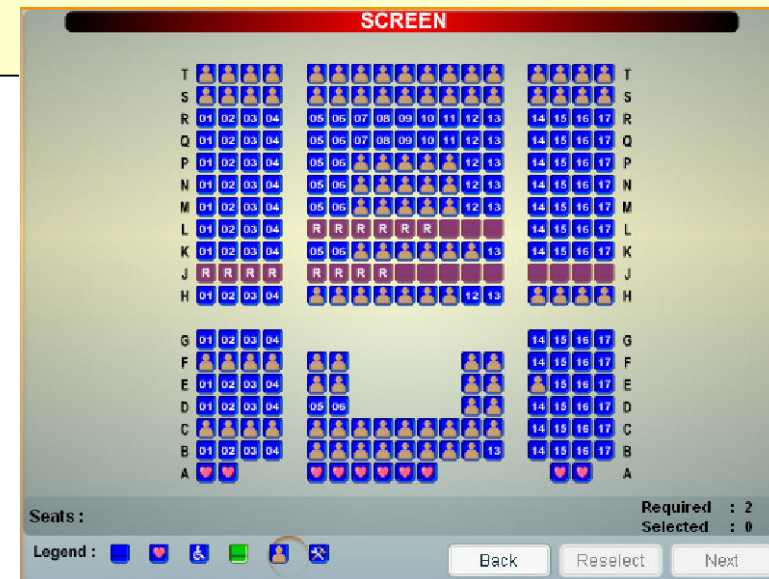
```
#include <stdio.h>
#define SIZE 10
int main(void)
{
    char ans1;
    int ans2, i;
    int seats[SIZE] = {0};
    while(1)
    {
        printf("좌석을 예약하시겠습니까?(y 또는n) ");
        scanf(" %c",&ans1);
```



실습: 극장 좌석 예약

```
if(ans1 == 'y')
{
    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]);
    printf("\n");
    printf("몇번째 좌석을 예약하시겠습니까);
    scanf("%d",&ans2);
```

현재 좌석 예약
상태 출력





실습: 극장 좌석 예약

```
if(ans2 <= 0 || ans2 > SIZE) {  
    printf("1부터 10사이의 숫자를 입력하세요\n");  
    continue;  
}  
if(seats[ans2-1] == 0) { // 예약되지 않았으면  
    seats[ans2-1] = 1;  
    printf("예약되었습니다.\n");  
}  
else // 이미 예약되었으면  
    printf("이미 예약된 자리입니다.\n");  
}  
else if(ans1 == 'n')  
    return 0;  
}  
return 0;  
}
```

예약 성공



도전문제

- 위의 프로그램에서는 한명만 예약할 수 있다. 하지만 극장에 혼자서 가는 경우는 드물다. 따라서 한번에 **2명**을 예약할 수 있도록 위의 프로그램을 변경하여 보자.





lab: 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.





실행 결과

1 2 3 4 5 6 7 8 9 10

28 81 60 83 67 10 66 97 37 94

최소값은 10입니다.

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



알고리즘

배열 `prices[]`의 원소를 난수로 초기화한다.

일단 첫 번째 원소를 최소값 `minium` 이라고 가정한다.

```
for(i=1; i<배열의 크기; i++)
```

```
    if ( prices[i] < minimum )
```

```
        minimum = prices[i]
```

반복이 종료되면 `minimum`에 최소값이 저장된다.



시습: 최소값 찾기

```
#include <stdio.h>
```

```
#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++){
```

```
        prices[i] = (rand()%100)+1;
```

```
        printf("%-3d ", prices[i]);
```

```
    }
```

```
    printf("\n\n");
```

물건의 가격
출력



시작: 최소값 찾기

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

첫 번째 배열 원소를 최소
값으로 가정

현재의 최소값보다
배열 원소가
작으면, 배열 원
소를 최소값으로
복사한다.



^ ^ ^ ^ V V V V V





도전문제

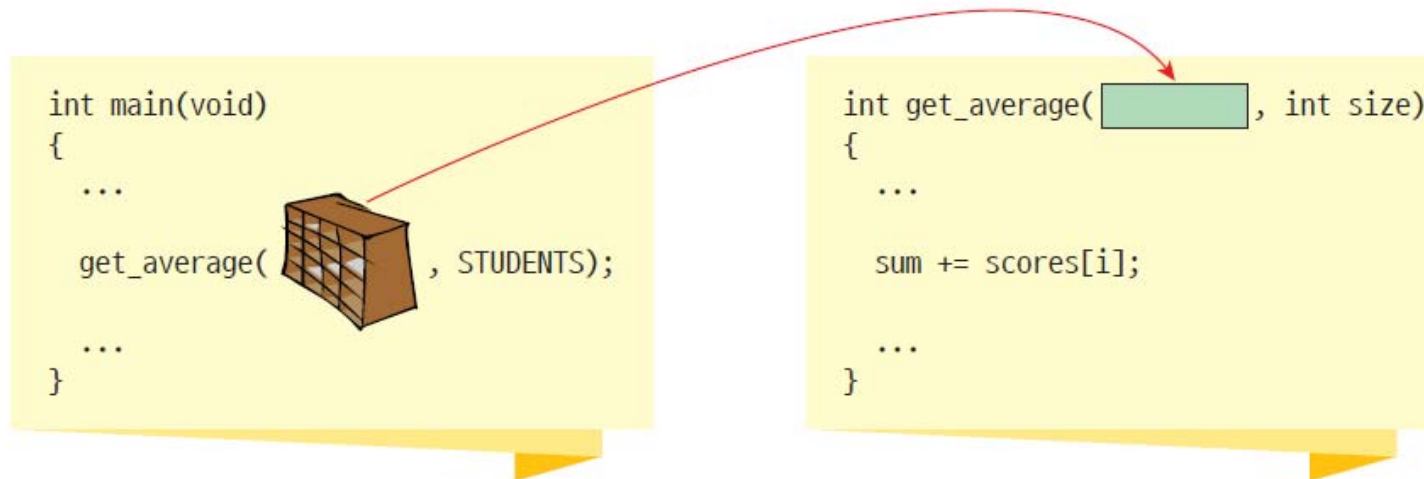
- 위의 프로그램에서는 최소값을 계산하였다. 이번에는 배열의 원소 중에서 최대값을 찾도록 변경하여 보자. 변수 이름도 적절하게 변경하라.





배열과 함수

- 배열의 경우에는 사본이 아닌 원본이 전달된다.





배열과 함수

```
#include <stdio.h>
#define STUDENTS 5
int get_average(int scores[], int n); // ①
```

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

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

배열이 인수인 경우,
참조에 의한 호출

배열의 원본이
score[]로 전달

```
int get_average(int scores[], int n) // ②
{
    int i;
    int sum = 0;

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



배열이 함수의 인수인 경우 1/2

```
#include <stdio.h>
```

```
#define SIZE 7
```

```
void square_array(int a[], int size);
```

```
void print_array(int a[], int size);
```

```
int main(void)
```

```
{
```

```
    int list[SIZE] = { 1, 2, 3, 4, 5, 6, 7 } ;
```

```
    print_array(list, SIZE);
```

```
    square_array(list, SIZE);
```

```
    print_array(list, SIZE);
```

```
    return 0;
```

```
}
```

배열은 원본이 전달된다. (인수 : 배열)



배열이 함수의 인수인 경우 2/2

```
void square_array(int a[], int size)
```

```
{  
    int i;  
  
    for(i = 0; i < size; i++)  
        a[i] = a[i] * a[i];  
}
```

```
void print_array(int a[], int size)
```

```
{  
    int i;  
  
    for(i = 0; i < size; i++)  
        printf("%3d ", a[i]);  
    printf("\n");  
}
```

배열의 원본이 a[]로 전달

```
1 2 3 4 5 6 7  
1 4 9 16 25 36 49
```



언변 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?



장간 점검

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





정렬이란?

- 정렬은 물건을 크기순으로 오름차순이나 내림차순으로 나열하는 것
- 정렬은 컴퓨터 공학분야에서 가장 기본적이고 중요한 알고리즘중의 하나





정렬이란?

- 정렬은 자료 탐색에 있어서 필수적이다.
(예) 만약 사전에서 단어들이 정렬이 안되어 있다면?





선택정렬(selection sort)

- 선택정렬(selection sort): 정렬이 안된 숫자들중에서 최소값을 선택하여 배열의 첫번째 요소와 교환

왼쪽 배열	오른쪽 배열	설명
()	(5,3,8,1,2,7)	초기상태
(1)	(5,3,8,2,7)	1선택
(1,2)	(5,3,8,7)	2선택
(1,2,3)	(5,8,7)	3선택
(1,2,3,5)	(8,7)	5선택
(1,2,3,5,7)	(8)	7선택
(1,2,3,5,7,8)	()	8선택



선택정렬(selection sort)

- 선택정렬(selection sort): 정렬이 안된 숫자들중에서 최소값을 선택하여 배열의 첫 번째 요소와 교환





선택 정렬 Selection Sort

```
#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;
    }
}
```

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

list[i]와 list[least]를 서로 교환



선택 정렬 선택 정렬

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

원래의 배열

3 2 9 7 1 4 8 0 6 5

정렬된 배열

0 1 2 3 4 5 6 7 8 9



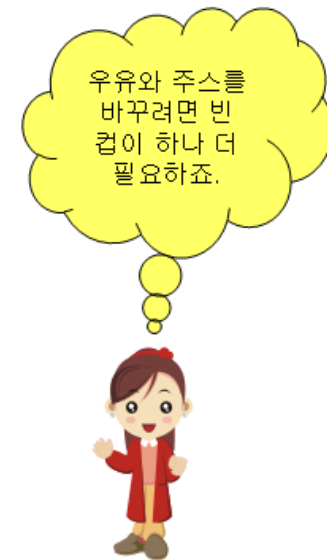
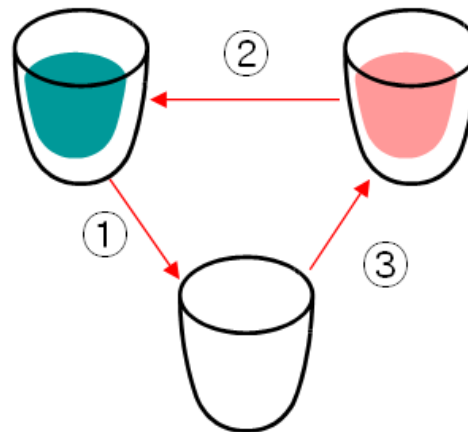
변수의 값을 서로 교환할 때

□ 다음과 같이 하면 안됨

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

□ 올바른 방법

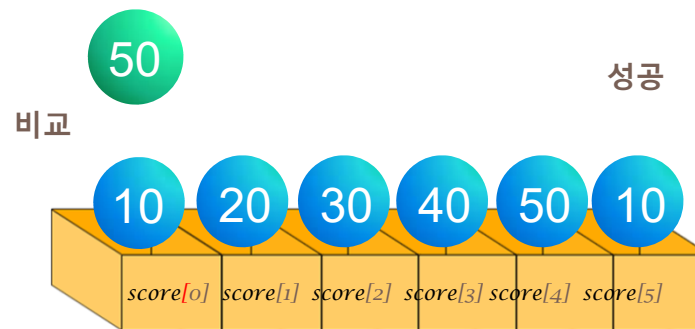
- ▣ `temp = list[i];`
- ▣ `list[i] = list[least];`
- ▣ `list[least] = temp;`





순차탐색

- 순차 탐색은 배열의 원소를 순서대로 하나씩 꺼내서 탐색 키와 비교하여 원하는 값을 찾아가는 방법





순차 탐색

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

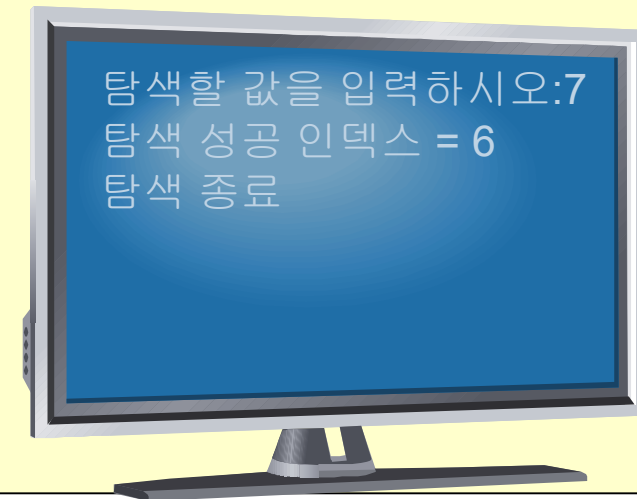
int main(void)
{
    int key, i;
    int list[SIZE] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };

    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;
}
```

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





이진 탐색

- 이진 탐색(binary search): 정렬된 배열의 중앙에 위치한 원소와 비교 되풀이





이진 탐색

```
#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;
}
```



이진 탐색

```
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] ) // 중간 원소보다 크다면
            low = middle + 1;          // 새로운 값으로 low 설정
        else
            high = middle - 1;         // 새로운 값으로 high 설정
    }
    return -1;
}
```



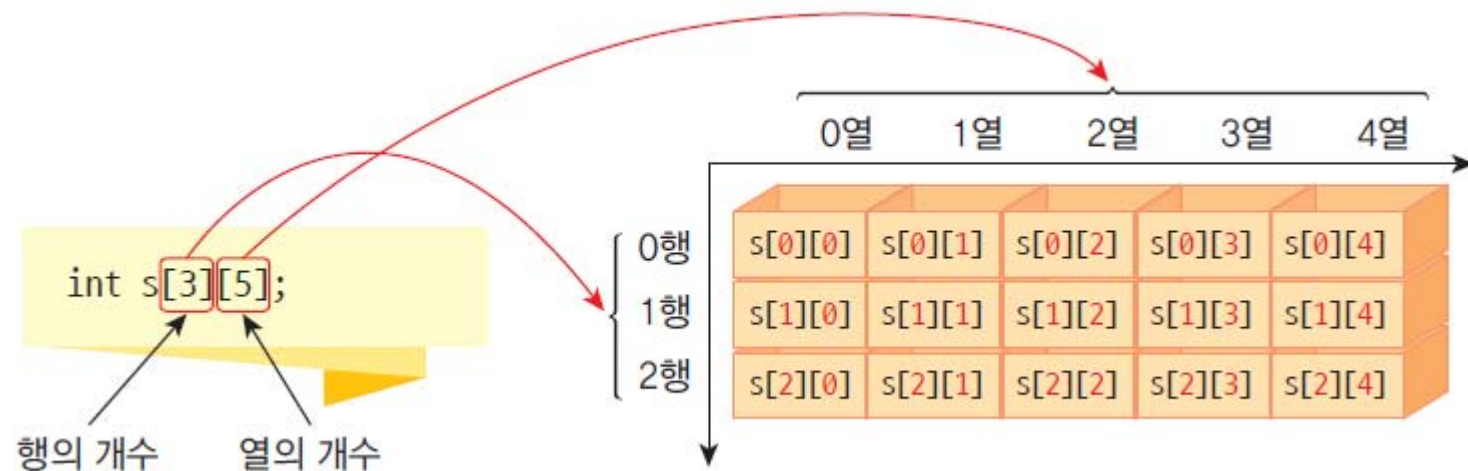
실행 결과





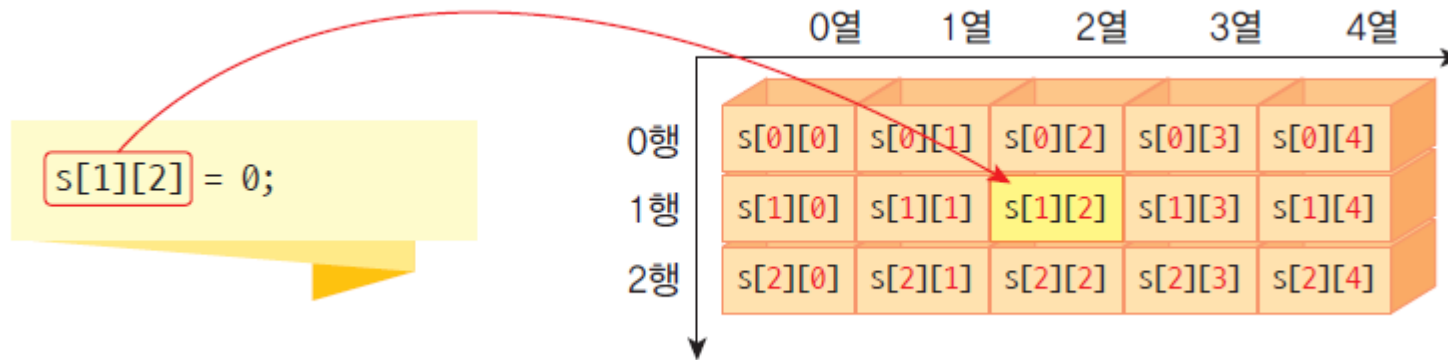
2차원 배열

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





2차원 배열에서 인덱스





2차원 배열의 활용

```
#include <stdio.h>
#include <stdlib.h>
#define ROWS 3
#define COLS 5
int main(void)
{
    int s[ROWS][COLS]; // 2차원 배열 선언
    int i, j; // 2개의 인덱스 변수
    for (i = 0; i < ROWS; i++)
        for (j = 0; j < COLS; j++)
            s[i][j] = rand() % 100;
    for (i = 0; i < ROWS; i++)
    {
        for (j = 0; j < COLS; j++)
            printf(" %02d ", s[i][j]);
        printf("\n");
    }
    return 0;
}
```





2차원 배열의 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  
};
```





2차원 배열의 초기화

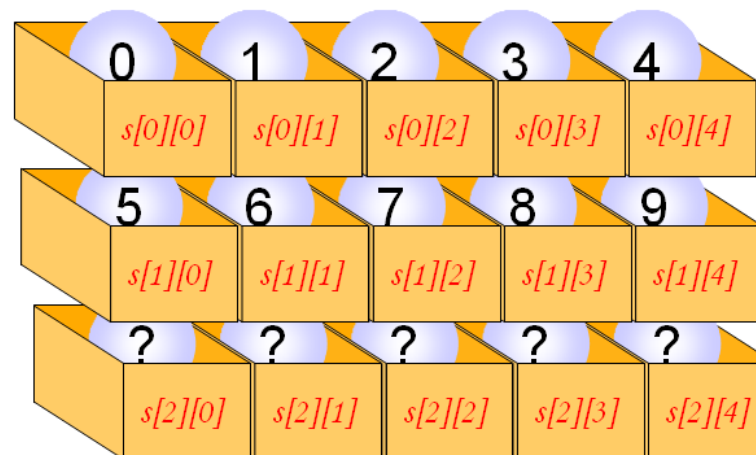
```
int s[ ][5] = {  
    { 0, 1, 2, 3, 4 }, // 첫 번째 행의 원소들의 초기값  
    { 10, 11, 12, 13, 14 }, // 두 번째 행의 원소들의 초기값  
    { 20, 21, 22, 23, 24 }, // 세 번째 행의 원소들의 초기값  
};
```





2차원 배열의 초기화

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





예제

- 학생들의 성적 기록표를 2차원 배열에 저장하고 각 학생의 최종 성적을 계산해보자.

학번	중간고사(30%)	기말고사(40%)	기말과제(20%)	퀴즈점수(10%)	결석횟수(감점)
1	87	98	80	76	3
2	99	89	90	90	0
3	65	68	50	49	0



2차원 배열의 초기화

```
#include <stdio.h>
#define ROWS 3
#define COLS 5

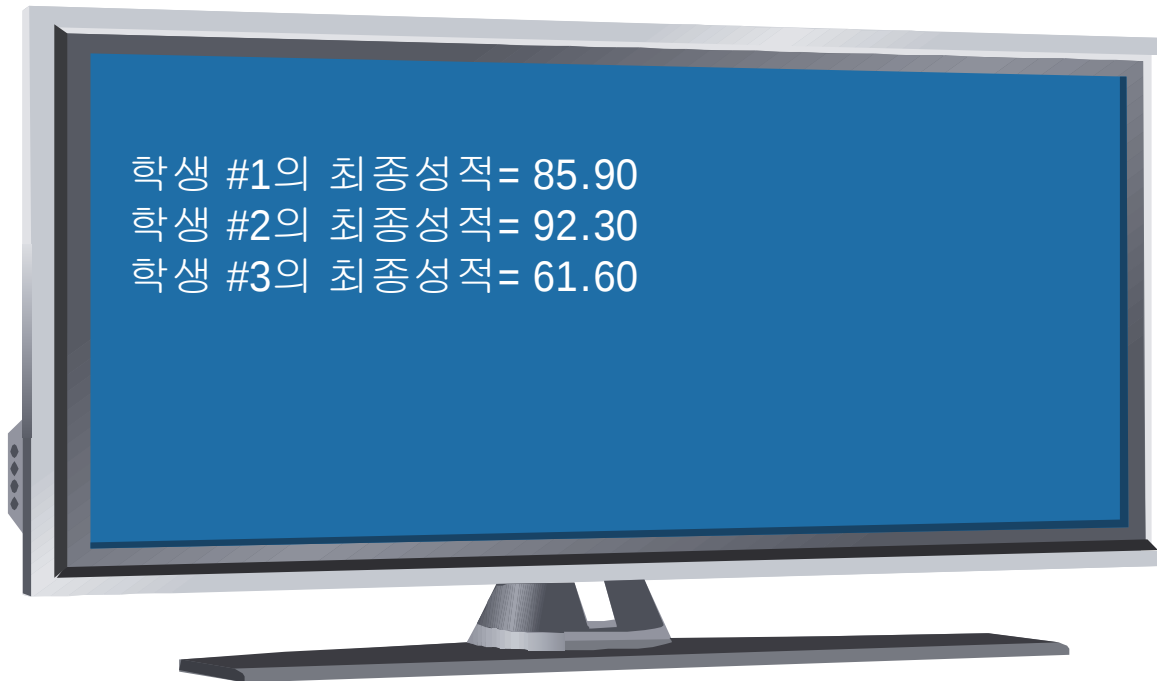
int main(void) {

    int a[ROWS][COLS] = { { 87, 98, 80, 76, 3 },
        { 99, 89, 90, 90, 0 },
        { 65, 68, 50, 49, 0 }
    };

    int i;
    for (i = 0; i < ROWS; i++) {
        double final_scores = a[i][0] * 0.3 + a[i][1] * 0.4 +
            a[i][2] * 0.2 + a[i][3] * 0.1 - a[i][4];
        printf("학생 #%i의 최종성적 = %10.2f \n", i + 1, final_scores);
    }
    return 0;
}
```



실행 결과





행렬

- 행렬(matrix)는 자연과학에서 많은 문제를 해결하는데 사용

$$A = \begin{bmatrix} 2 & 3 & 0 \\ 8 & 9 & 1 \\ 7 & 0 & 5 \end{bmatrix} \quad B = \begin{bmatrix} 0 & 0 & 7 & 0 & 0 \\ 9 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \\ 6 & 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 2 & 0 & 0 \end{bmatrix}$$

Mathematics - ELEMENTARY MATRIX OPERATIONS

OPERATION: MULTIPLY EACH ELEMENT IN 2ND Row by 7:

$A = \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix}$

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

2) PREMULT $\begin{bmatrix} 1 & 0 \\ 0 & 7 \end{bmatrix} \begin{bmatrix} 0 & 1 & 2 \\ 3 & 4 & 5 \end{bmatrix} \Rightarrow \begin{bmatrix} 1(0)+0(3) & 1(1)+0(4) & 1(2)+0(5) \\ 0(0)+7(3) & 0(1)+7(4) & 0(2)+7(5) \end{bmatrix}$



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];
```



다차원 배열을 이용한 행렬의 표현

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

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





2차원 배열을 함수로 전달하기

```
#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;
}
```



2차원 배열을 함수로 전달하기

```
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;
}
```



정간 점검

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



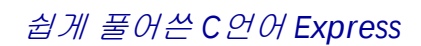


lab: 영상 처리

- 디지털 영상은 픽셀들의 2차원 배열이라 할 수 있다.

```
int image[8][16] = {  
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },  
    { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },  
    { 1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1 },  
    { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },  
    { 1,0,0,0,0,0,0,0,0,0,0,0,0,1,1,1 },  
    { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
```







```
#include <stdio.h>

void display(int image[8][16])
{
    for (int r = 0; r < 8; r++) {
        for (int c = 0; c < 16; c++) {
            if (image[r][c] == 0)
                printf("*");
            else
                printf("_");
        }
        printf("\n");
    }
}

void inverse(int img[8][16])
{
    for (int r = 0; r < 8; r++) {
        for (int c = 0; c < 16; c++) {
            if (img[r][c] == 0)
                img[r][c] = 1;
            else
                img[r][c] = 0;
        }
    }
}
```



```
int main(void)
{
    int image[8][16] = {
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,1,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,1,1,1,1,1,1,1,1,1,1,1 },
        { 1,1,1,0,0,0,1,1,0,0,1,1,1,1,1,1 },
        { 1,1,0,0,0,0,0,0,0,0,1,1,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,1,1,1,1,1 },
        { 1,0,0,0,0,0,0,0,0,0,0,0,1,1,1,1 },
        { 1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1 } };
    printf("변환전 이미지\n");
    display(image);
    inverse(image);
    printf("\n\n변환후 이미지\n");
    display(image);
    return 0;
}
```



mini project: tic-tac-toe

- tic-tac-toe 게임은 2명의 경기자가 오른쪽과 같은 보드를 이용하여서 번갈아가며 O와 X를 놓는 게임이다.
- 같은 글자가 가로, 세로, 혹은 대각선 상에 놓이면 이기게 된다.





실행 결과

```
C:\WINDOWS\system32\cmd.exe
(x, y) 좌표: 0 0
X
|
|
|
|
|
|
(x, y) 좌표: 1 1
X
|
| 0
|
|
|
(x, y) 좌표:
```



알고리즘

보드를 초기화한다.

`while(1)`

 보드를 화면에 출력한다.

 사용자로부터 좌표 `x, y`를 받는다.

 if (`board[x][y]`가 비어 있으면)

 if(현재 경기자가 'X' 이면)

`board[x][y] = 'X'`

 else

`board[x][y] = 'O'`

 else

 오류 메시지를 출력한다



```
#include <stdio.h>
int main(void)
{
    char board[3][3];
    int x, y, k, i;
    // 보드를 초기화한다.
    for (x = 0; x < 3; x++)
        for (y = 0; y < 3; y++) board[x][y] = ' ';

    // 사용자로부터 위치를 받아서 보드에 표시한다.
    for (k = 0; k < 9; k++) {
        printf("(x, y) 좌표: ");
        scanf(" %d %d", &x, &y);
        board[x][y] = (k % 2 == 0) ? 'X' : 'O'; // 순번에 따라 'X', 'O'중
```



```
// 보드를 화면에 그린다.  
for (i = 0; i < 3; i++) {  
    printf("---|---|---\n");  
    printf("%c | %c | %c \n", board[i][0], board[i][1],  
board[i][2]);  
}  
printf("---|---|---\n");  
}  
return 0;  
}
```



도전문제

- (1) 위의 코드를 실행하면 상대방이 놓은 곳에 다시 놓을 수 있다. 이것을 방지하는 코드를 추가하라.
- (2) 보드를 분석하여서 게임이 종료되었는지를 검사하는 함수를 추가하라.
- (3) 컴퓨터가 자동으로 다음 수를 결정하도록 프로그램을 변경하라. 가장 간단한 알고리즘을 사용한다. 예를 들면 비어 있는 첫 번째 좌표에 놓는다.





Q & A

