


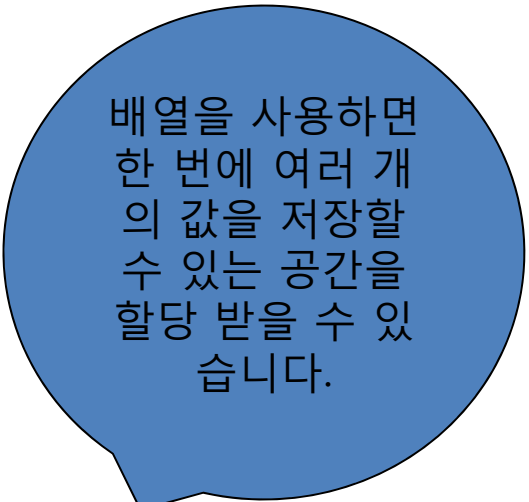
Computer Engineering Programming 2

제10장 배열

Lecturer: JUNBEOM YOO
jbyoo@konkuk.ac.kr

이번 장에서 학습할 내용

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



배열을 사용하면
한 번에 여러 개
의 값을 저장할
수 있는 공간을
할당 받을 수 있
습니다.



배열의 필요성

- 학생이 10명이 있고 이들의 평균 성적을 계산한다고 가정하자.

개별 변수를 사용하는 방법은 학생 수가 많아지면 번거로워집니다..



방법 #1: 개별 변수 사용

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

방법 #2: 배열 사용

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

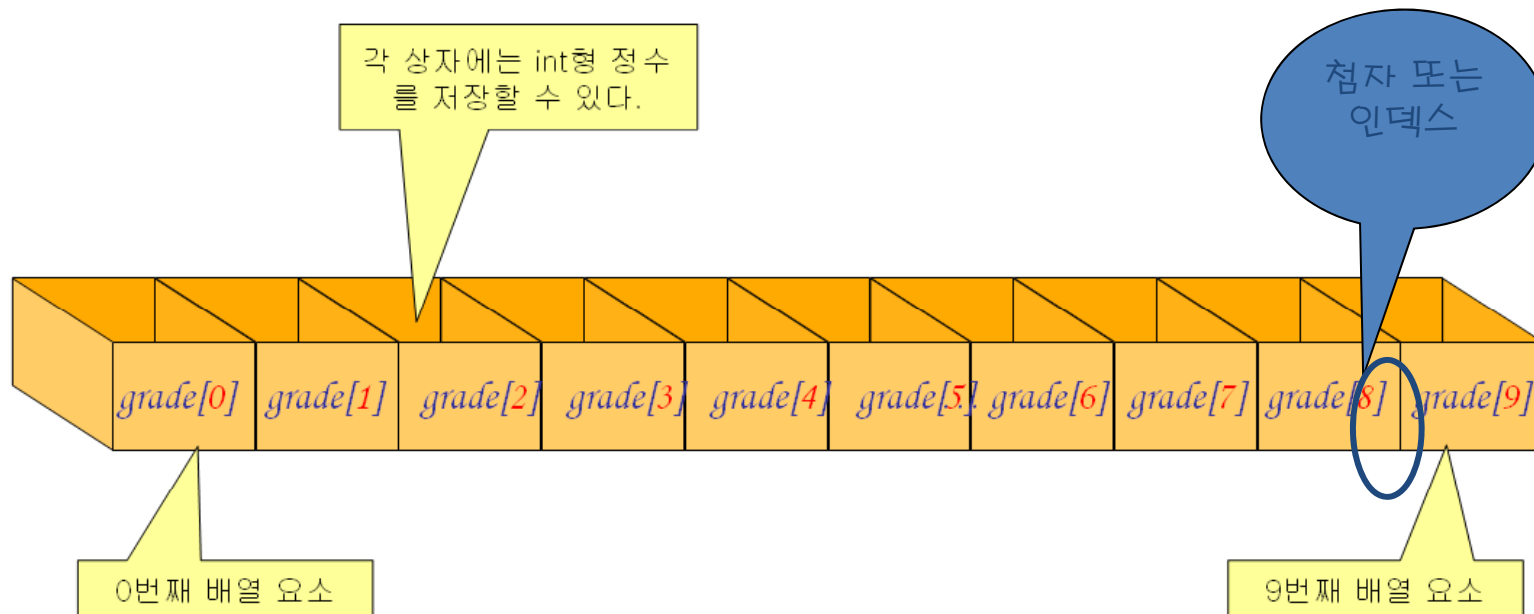
배열이란?

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

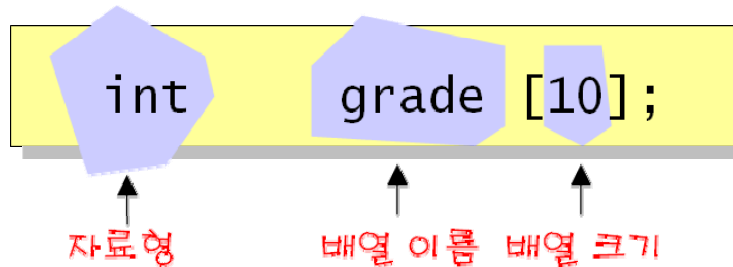


배열 원소와 인덱스

- *인덱스(index)*: 배열 원소의 번호



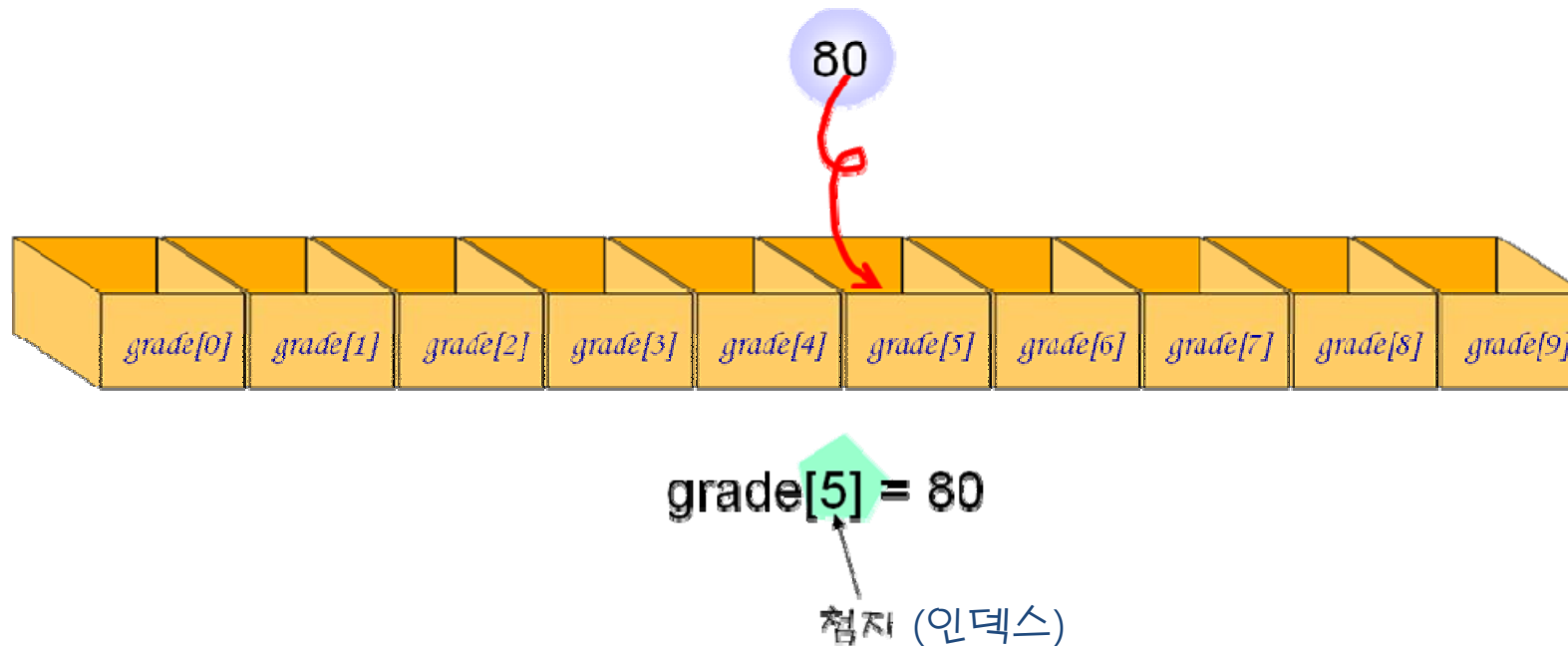
배열의 선언



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

```
int score[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
char src[10], dst[10];   // 2개의 문자형 배열을 동시에 선언
int index, days[7];      // 일반 변수와 배열을 동시에 선언
```

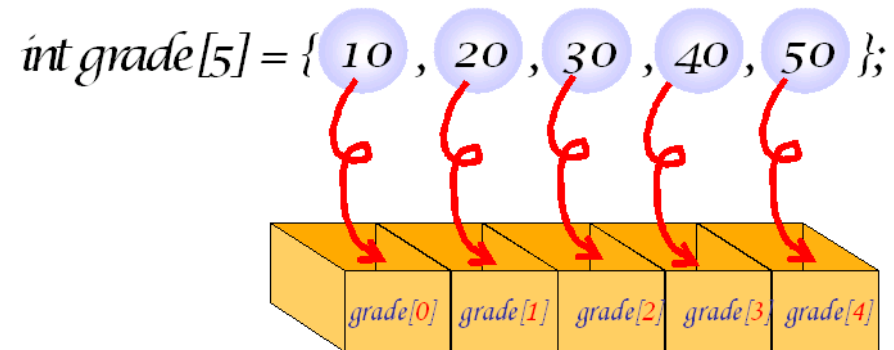
배열 원소 접근



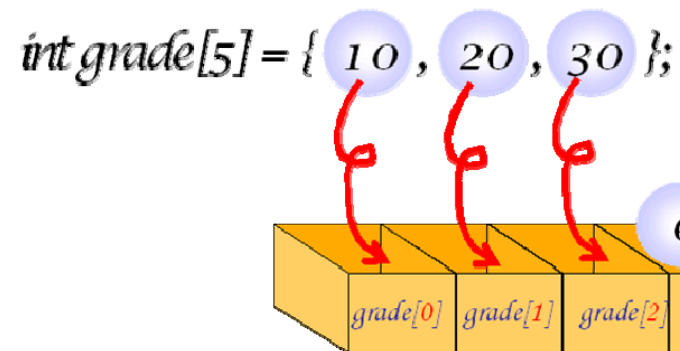
```
grade[5] = 80;  
grade[1] = grade[0];  
grade[i] = 100;    // i는 정수 변수  
grade[i+2] = 100; // 수식이 인덱스가 된다.  
grade[index[3]] = 100; // index[]는 정수 배열
```

배열의 초기화

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



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

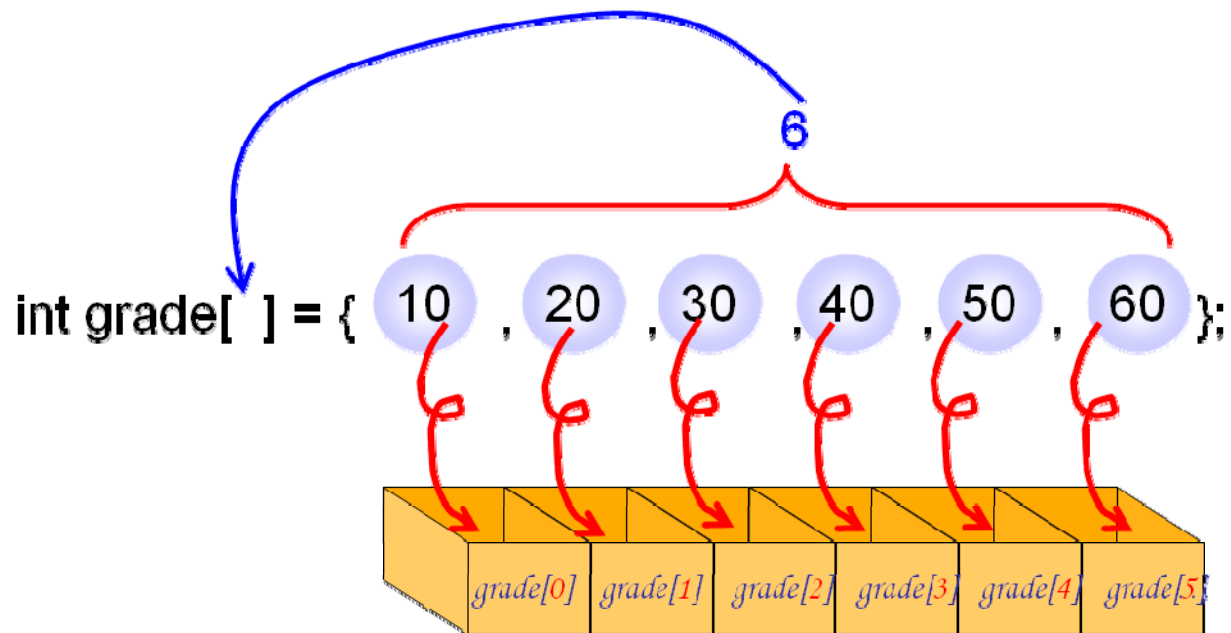


초기값을 일부만 주면 나머지 원소들은 0으로 초기화됩니다.



배열의 초기화

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



배열 선언 예제



```
#include <stdio.h>
int main(void)
{
    int grade[10];
    int i;

    for(i = 0; i < 10; i++)
        grade[i] = 0;

    printf("=====\n");
    printf("인덱스   값\n");
    printf("=====\n");

    for(i = 0; i < 10; i++)
        printf("%5d  %5d\n", i, grade[i]);

    return 0;
}
```



인덱스	값
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

배열 초기화 예제



```
#include <stdio.h>
int main(void)
{
    int grade[10] = { 31, 63, 62, 87, 14, 25, 92, 70, 75, 53 };
    int i;

    printf("=====\n");
    printf("인덱스   값\n");
    printf("=====\n");

    for(i = 0; i < 10; i++)
        printf("%5d  %5d\n", i, grade[i]);

    return 0;
}
```



=====	
인덱스	값
=====	
0	31
1	63
2	62
3	87
4	14
5	25
6	92
7	70
8	75
9	53

배열 원소 참조 예제



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

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

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

    printf("=====\n");
    printf("인덱스   값\n");
    printf("=====\n");

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



```
=====  
인덱스   값  
=====  
0       41  
1       67  
2       34  
3        0  
4       69  
5       24  
6       78  
7       58  
8       62  
9       64
```

배열 원소 사용 예제



```
#include <stdio.h>

#define STUDENTS 5

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

    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;
}
```



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

잘못된 인덱스로 접근하는 경우



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

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

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

    return 0;
}
```



```
array[0]    1
array[1]    2
array[2]    3
array[3]    4
array[4]    5
array[5]    1245120
```

배열의 복사

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

잘못된 방법

```
score = grade; // 컴파일 오류!
```



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

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

```
    int i;  
    int a[SIZE] = {1, 2, 3, 4, 5};  
    int b[SIZE];
```

```
    for(i = 0; i < SIZE; i++)  
        b[i] = a[i];
```

올바른 방법

```
    return 0;
```

```
}
```

배열의 비교



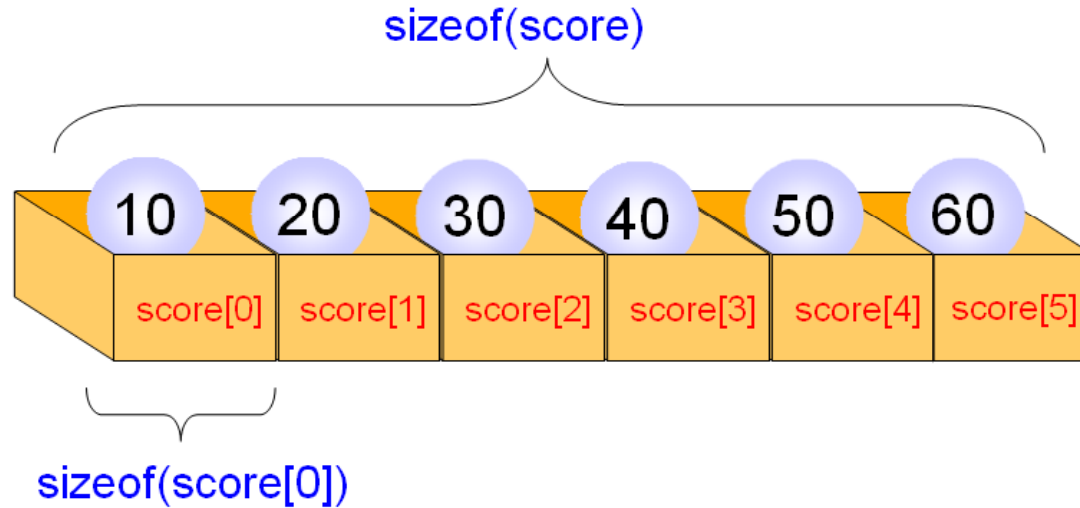
```
#include <stdio.h>
#define SIZE 5

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 올바른지 않은 배열 비교
        printf("잘못된 결과입니다.\n");
    else
        printf("잘못된 결과입니다.\n");

    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;
}
```


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]);
```

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 입력 받는 루프
    {
        printf("정수를 입력하시오:");
        scanf("%d", &data[i]);
    }

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



```
정수를 입력하시오:10
정수를 입력하시오:20
정수를 입력하시오:30
정수를 입력하시오:40
정수를 입력하시오:50
50
40
30
20
10
```

예제



```
#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;
}
```



```
번호 0: *****
번호 1: *****
번호 2: *****
번호 3: *****
번호 4: *****
```

최소값 탐색



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

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

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

    min = grade[0];

    for(i = 1; i < SIZE; i++)
    {
        if( grade[i] < min )
            min = grade[i];
    }

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



숫자를 입력하시오: 50
숫자를 입력하시오: 40
숫자를 입력하시오: 30
숫자를 입력하시오: 20
숫자를 입력하시오: 10
숫자를 입력하시오: 20
숫자를 입력하시오: 30
숫자를 입력하시오: 40
숫자를 입력하시오: 60
숫자를 입력하시오: 70
최소값은 10입니다.

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;
}
```



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
...	
98	0
99	1
100	2

주사위면 빈도 계산



```
#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;
}
```



면	빈도
0	1657
1	1679
2	1656
3	1694
4	1652
5	1662

배열과 함수



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

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) // ②
{
    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

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);
void square_element(int e);

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

    print_array(list, SIZE);
    square_array(list, SIZE); // 배열은 원본이 전달된다.
    print_array(list, SIZE);

    printf("%3d\n", list[6]);
    square_element(list[6]); // 배열 요소는 복사본이 전달된다.
    printf("%3d\n", list[6]);

    return 0;
}
```


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 square_element(int e)
{
    e = e * e;
}
void print_array(int a[], int size)
{
    int i;

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



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

원본 배열의 변경을 금지하는 방법



```
#include <stdio.h>
#define SIZE 20
```

```
void copy_array(char dest[], const char src[], int count);
```

```
int main(void)
```

```
{
```

```
    char s[SIZE] = { 'H', 'E', 'L', 'L', 'O', '\0' };
    char d[SIZE];
```

```
    copy_array(d, s, SIZE);
    printf("%s\n", d);
    return 0;
```

```
}
```

```
void copy_array(char dest[], const char src[], int size)
```

```
{
```

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

배열 원본의 변경 금지



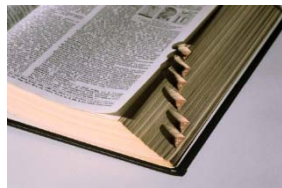
정렬이란?

- 정렬은 물건을 크기 순으로 오름차순이나 내림차순으로 나열하는 것



- 정렬은 컴퓨터 공학분야에서 가장 기본적이고 중요한 알고리즘중의 하나

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



비교	제조사	모델명	요약설명	최저가	업체수	출시
<input type="checkbox"/>	ROLLEI	D-41com	410만화소(0.56")/1.8"LCD/3배줌/연사/CF카드	320,000	74	02년
<input type="checkbox"/>	카시오	QV-R40	413만화소(0.56")/1.6"LCD/3배줌/동영상/히스토그램/앨범기능/SD,MMC카드	344,000	73	03년
<input type="checkbox"/>	파나소닉	DMC-LC43	423만화소(0.4")/1.5"LCD/3배줌/동영상+녹음/연사/SD,MMC카드	348,000	36	03년
<input type="checkbox"/>	현대	DC-4311	400만화소(0.56")/1.6"LCD/3배줌/동영상/SD,MMC카드	350,000	7	03년
<input type="checkbox"/>	삼성테크윈	Digimax420	410만화소(0.56")/1.5"LCD/3배줌/동영상+녹음/음성메모/한글/SD카드	353,000	17	03년
<input type="checkbox"/>	니콘	Coolpix4300	413만화소(0.56")/1.5"LCD/3배줌/동영상/연사/CF카드*hot4	356,800	79	02년
<input type="checkbox"/>	올림푸스	뮤-20 Digital	423만화소(0.4")/1.5"LCD/3배줌/동영상/연사/생활방수/xD카드	359,000	33	03년
<input type="checkbox"/>	코닥	LS-443(Dock포함)	420만화소/1.8"LCD/3배줌/동영상+녹음/SD,MMC카드/Dock시스템	365,000	39	02년
<input type="checkbox"/>	올림푸스	C-450Z	423만화소(0.4")/1.8"LCD/3배줌/동영상/연사/xD카드	366,000	38	03년
<input type="checkbox"/>	올림푸스	X-1	430만화소/1.5"LCD/3배줌/동영상/연사/xD카드	367,000	19	03년
<input type="checkbox"/>	미놀타	DIMAGE-F100	413만화소(0.56")/1.5"LCD/3배줌/동영상+녹음/음성메모/동체 추적 AF/연사/SD,MMC카드	373,000	18	02년
<input type="checkbox"/>	삼성테크윈	Digimax410	410만화소(0.56")/1.6"LCD/3배줌/동영상+녹음/음성메모/한글/CF카드	374,000	4	02년

선택정렬(selection sort)

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



선택 정렬 1/2



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

void selection_sort(int list[], int n);
void print_list(int list[], int n);

int main(void)
{
    int grade[SIZE] = { 3, 2, 9, 7, 1, 4, 8, 0, 6, 5 };

    // 원래의 배열 출력
    printf("원래의 배열\n");
    print_list(grade, SIZE);

    selection_sort(grade, SIZE);

    // 정렬된 배열 출력
    printf("정렬된 배열\n");
    print_list(grade, SIZE);

    return 0;
}
```

선택 정렬 2/2

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

void selection_sort(int list[], int n)
{
    int i, j, temp, least;

    for(i = 0; i < n-1; i++)
    {
        least = i;

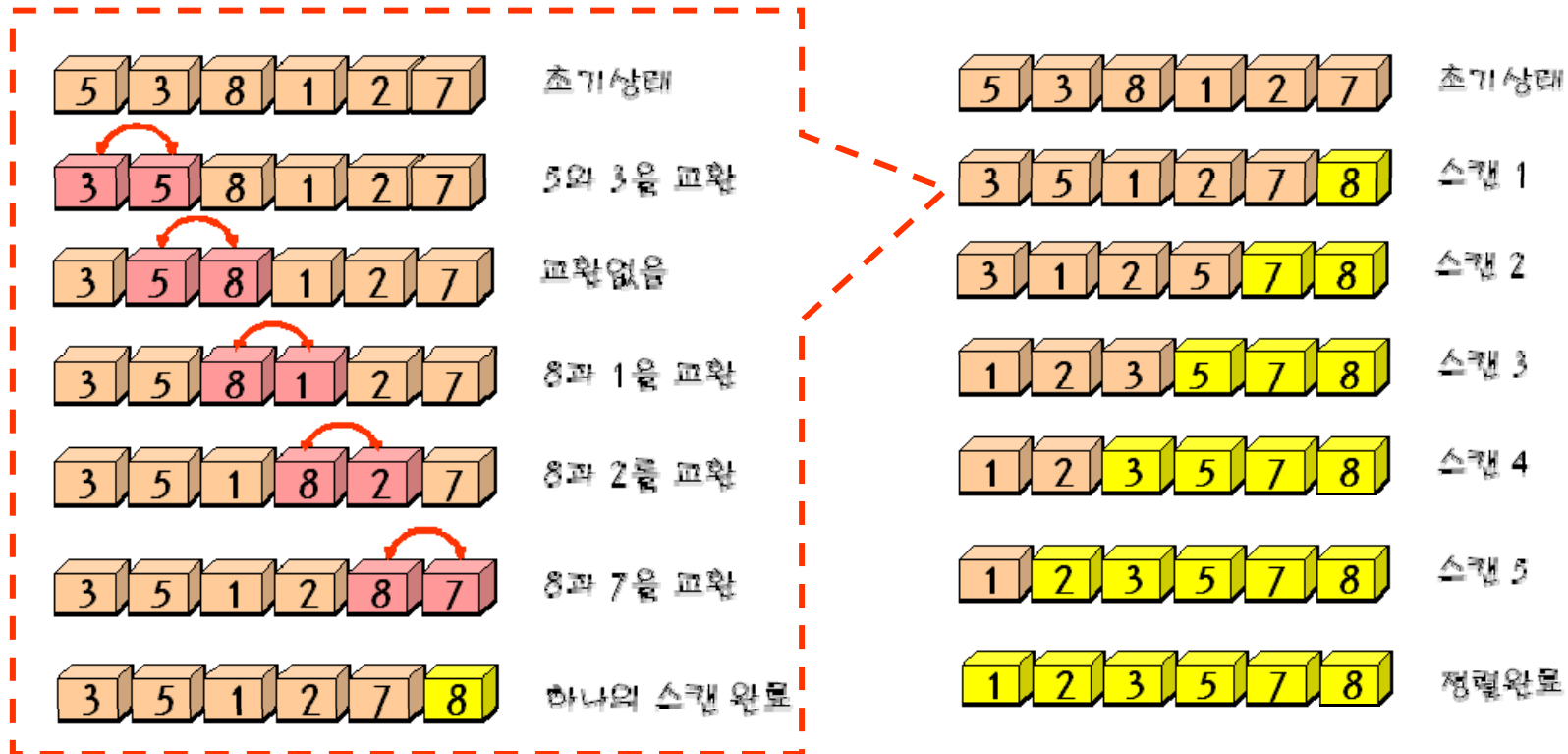
        for(j = i + 1; j < n; j++) // 최소값 탐색
            if(list[j] < list[least])
                least = j;
        // i번째 원소와 least 위치의 원소를 교환
        temp = list[i];
        list[i] = list[least];
        list[least] = temp;
    }
}
```



원래의 배열
3 2 9 7 1 4 8 0 6 5
정렬된 배열
0 1 2 3 4 5 6 7 8 9

버블정렬(bubble sort)

- 인접한 레코드가 순서대로 되어 있지 않으면 교환
- 전체가 정렬될 때까지 비교/교환 계속



버블 정렬



```
void bubble_sort(int list[], int n)
{
    int i, scan, temp;

    // 스캔 회수를 제어하기 위한 루프
    for(scan = 0; scan < n-1; scan++)
    {
        // 인접값 비교 회수를 제어하기 위한 루프
        for(i = 0; i < n-1; i++)
        {
            // 인접값 비교 및 교환
            if( list[i] > list[i+1] )
            {
                temp = list[i];
                list[i] = list[i+1];
                list[i+1] = temp;
            }
        }
    }
}
```


순차 탐색



```
#include <stdio.h>
#define SIZE 6

int seq_search(int list[], int n, int key);

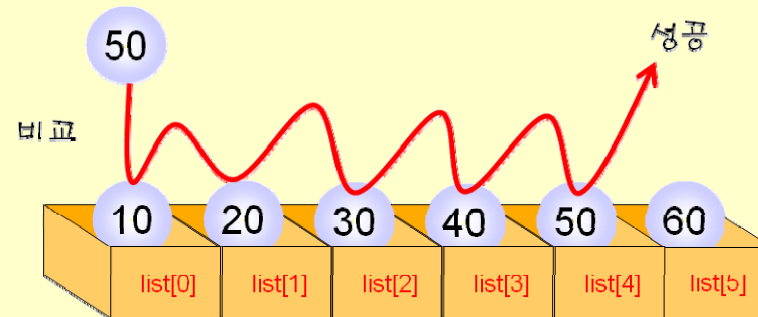
int main(void)
{
    int key;
    int grade[SIZE] = { 10, 20, 30, 40, 50, 60 };

    printf("탐색할 값을 입력하시오:");
    scanf("%d", &key);
    printf("탐색 결과 = %d\n", seq_search(grade, SIZE, key));

    return 0;
}

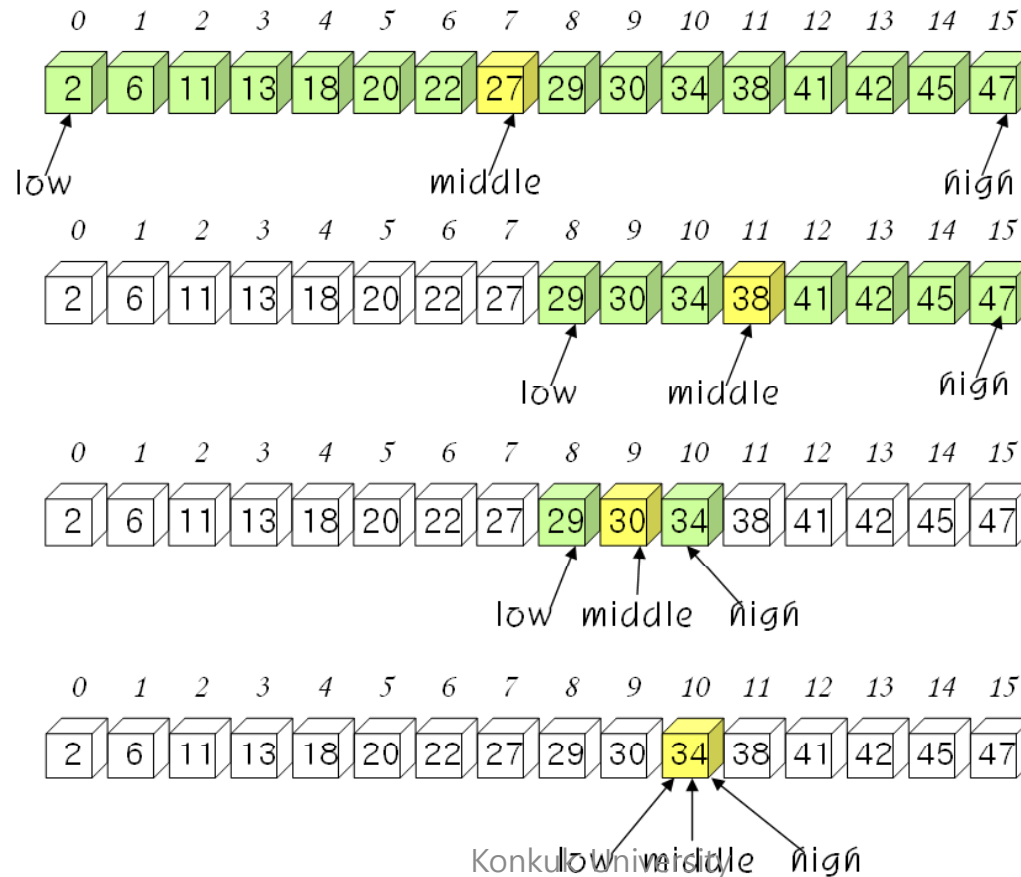
int seq_search(int list[], int n, int key)
{
    int i;

    for(i = 0; i < SIZE; i++)
        if(list[i] == key)
            return i; // 탐색이 성공하면 인덱스 반환
    return -1; // 탐색이 실패하면 -1 반환
}
```



이진 탐색

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



이진 탐색



```
int binary_search(int list[], int n, int key)
{
    int low, high, middle;

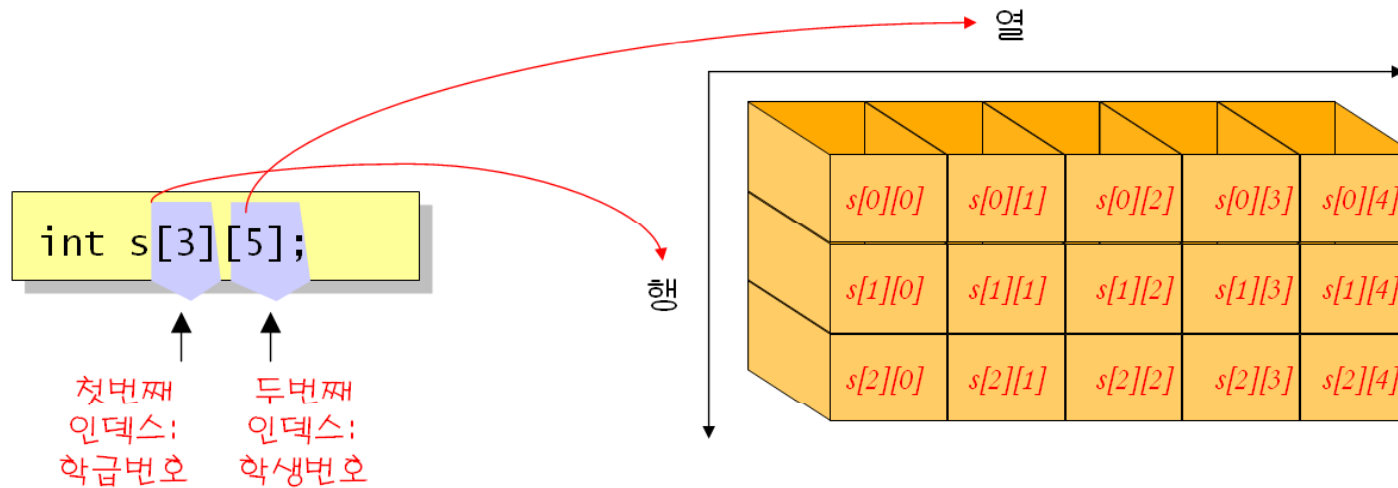
    low = 0;
    high = n-1;

    while( low <= high ){ // 아직 숫자들이 남아있으면
        middle = (low + high)/2; // 중간 요소 결정
        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;
}
```

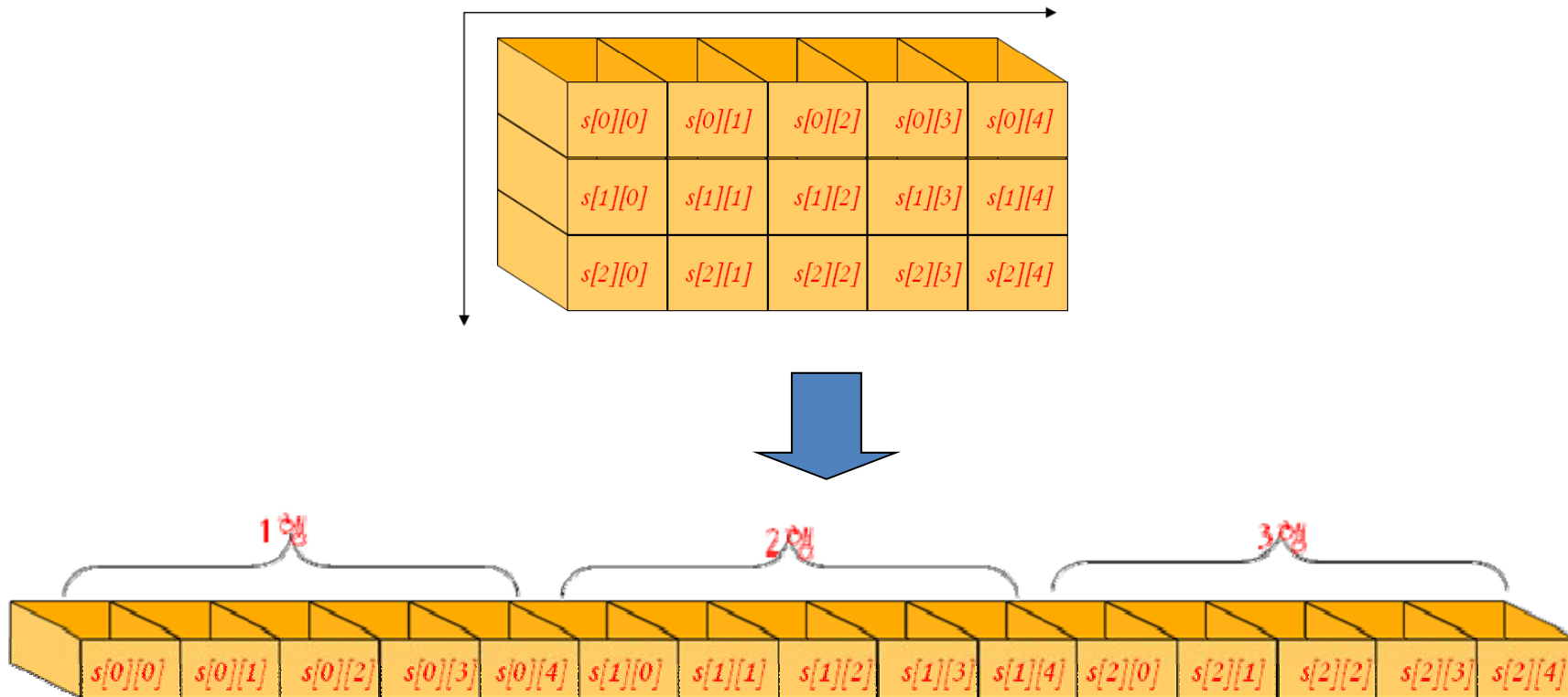
2차원 배열

```
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

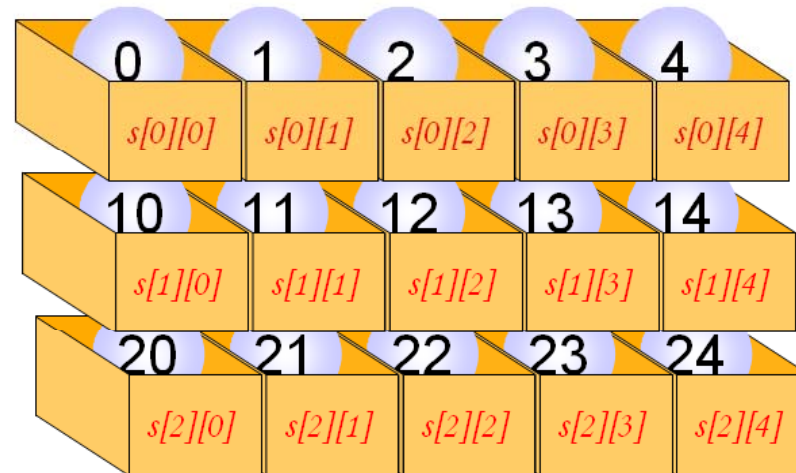
    for(i=0;i<3;i++)
        for(j=0;j<5;j++)
            s[i][j] = value++;

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

    return 0;
}
```

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  
};
```



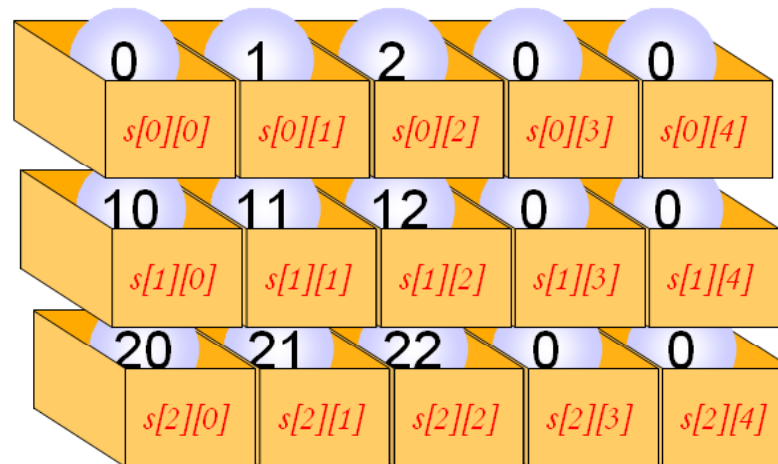
2차원 배열의 초기화

```
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  
};
```



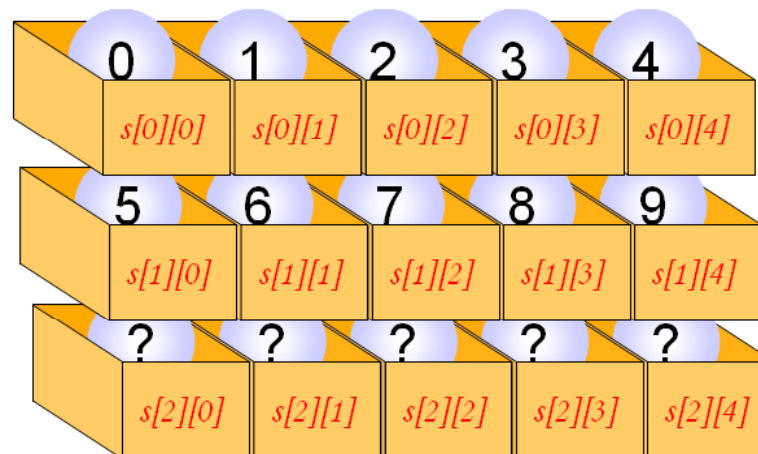
2차원 배열의 초기화

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

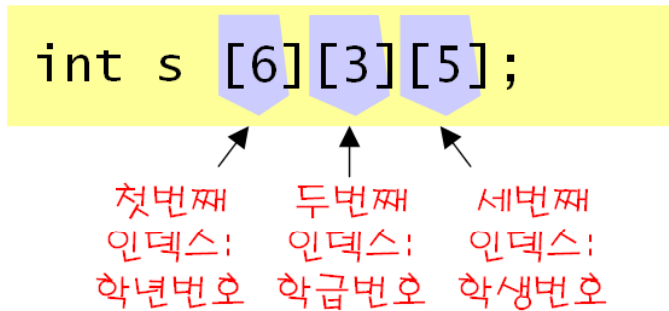


2차원 배열의 초기화

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



3차원 배열



```
#include <stdio.h>

int main(void)
{
    int s[3][3][3];    // 3차원 배열 선언
    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++;

    return 0;
}
```

다차원 배열 인수



```
#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;
}
```

첫번째 인덱스의 크기는 적지 않아도 된다.

다차원 배열 예제



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



학급 0의 평균 성적 = 2
학급 1의 평균 성적 = 12
학급 2의 평균 성적 = 22
전체 학생들의 평균 성적 = 12

```
int main(void)
{
    int s[CLASSES][STUDENTS] = {
        { 0, 1, 2, 3, 4 }, // 첫번째 행의 원소들의 초기값
        { 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;
}
```

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;
    // 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;
}
```



```
3 3 0
9 9 1
8 0 5
```

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

Q & A

