



C 프로그래밍 및 실습

C Programming

이지민

CHAP 07 배열

목차



배열

- 배열의 기본
 - 배열의 개념
 - 배열의 선언
 - 배열의 초기화
 - 배열의 사용



다차원 배열

- 다차원 배열의 개념
- 2차원 배열의 선언 및 사용
- 2차원 배열의 초기화

배열_배열의 기본

❖ 5개의 정수를 입력받으려면 5개의 정수형 변수를 선언해야 한다.

이름이 다른 변수를 여러 개 사용하는 경우

```
int main(void)
{
    int a, b, c, d, e;
    int sum = 0;

    scanf("%d", &a);
    sum += a;
    scanf("%d", &b);
    sum += b;
    scanf("%d", &c);
    sum += c;
    scanf("%d", &d);
    sum += d;
    scanf("%d", &e);
    sum += e;

    printf("sum = %d\n", sum);
}
```

이름이 다른 변수 5개를 선언한다.

변수의 이름이 다르기 때문에 반복문을 사용할 수 없다.

코드가 길고 복잡하다.

배열을 사용하는 경우

```
int main(void)
{
    int arr[5];
    int sum = 0;
    int i;

    for (i = 0; i < 5; i++)
    {
        scanf("%d", &arr[i]);
        sum += arr[i];
    }

    printf("sum = %d\n", sum);
}
```

크기가 5인 배열을 선언한다.

반복문을 사용 할 수 있다.

배열의 원소에 대하여 같은 코드를 수행한다.

배열의 원소들은 같은 이름을 사용하며 인덱스로 구분해서 사용한다.

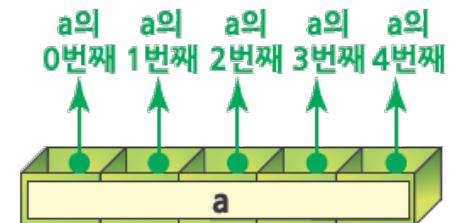
배열_배열의 기본

❖ 배열

- 같은 데이터형의 변수를 메모리에 연속적으로 할당하고 같은 이름으로 사용하는 방법을 제공
- 배열의 원소(element)
 - 배열 안에 들어가는 변수 하나하나
 - 개별적인 변수인 것처럼 사용
- 인덱스(index)
 - 배열의 각 원소를 구분하기 위한 번호
 - 항상 0부터 시작한다.
 - arr[0], arr[1], ...과 같은 방식으로 배열의 원소를 구분
- 배열의 각 원소도 변수이다.
 - 배열의 각 원소는 메모리에 연속적으로 할당됨



서로 다른 이름의 변수를
사용하는 경우



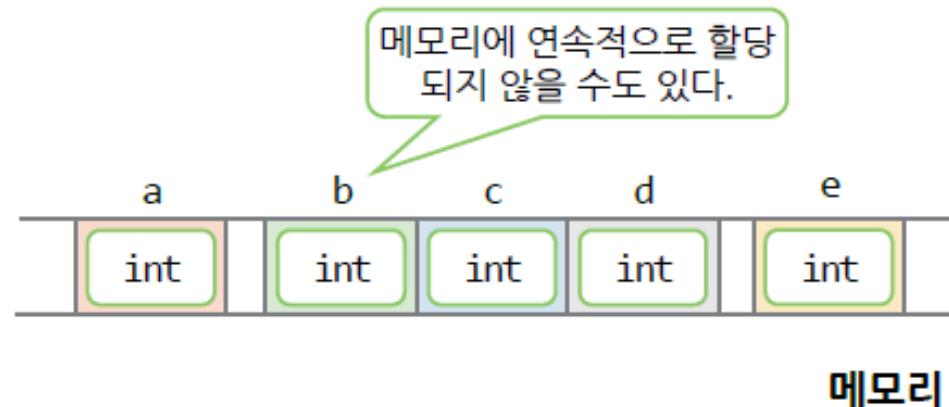
배열을 사용하는 경우

배열_배열의 개념

❖ 배열의 이름은 배열 전체에 대한 이름이 된다.

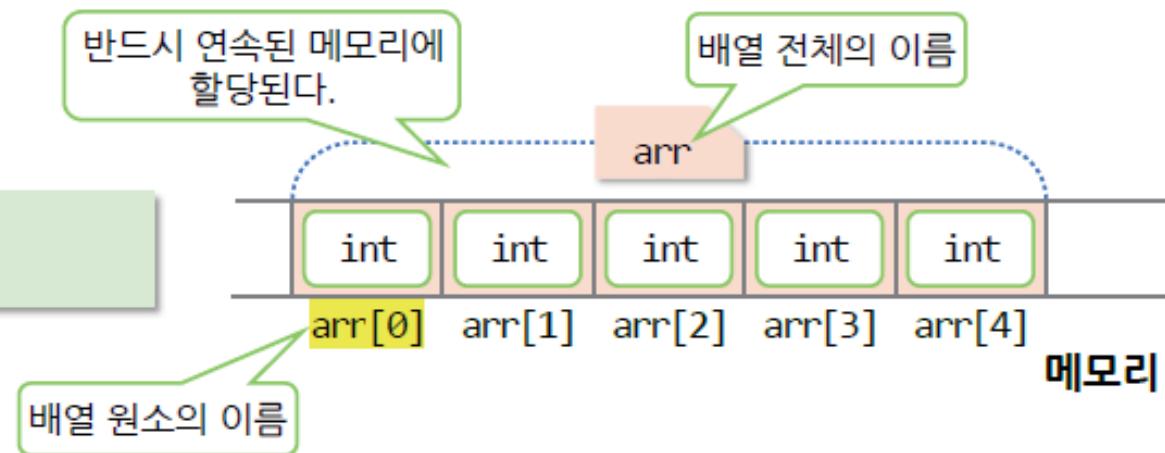
서로 다른 변수로 선언하는 경우

```
int a, b, c, d, e;
```



배열로 선언하는 경우

```
int arr[5];
```



배열_배열의 기본

❖ 배열의 선언(1/3)

- 배열을 선언하려면 배열의 데이터형과 배열명, 배열의 크기가 필요하다.

형식 데이터형 배열명[크기];

사용예

```
int num[5];
double data[100];
char name[32];
```

- 변수의 메모리 할당

int개 5개를 연속된
메모리에 할당한다.

```
int arr[5];
```

arr

int 5개만큼의 크기
 $4 \times 5 = 20$ 바이트

int int int int int

arr[0] arr[1] arr[2] arr[3] arr[4]

int 1개 크기
(4바이트)

메모리

배열_배열의 기본

❖ 배열의 선언(2/3)

- 배열의 크기는 반드시 0보다 큰 정수형 상수로 지정해야 한다.

```
int num[0];           // 배열의 크기는 0이 될 수 없다.  
int size = 100;  
double data[size];   // 배열의 크기를 변수로 지정할 수 없다.  
char name[];         // 크기를 지정해야 한다.
```

컴파일 에러

- 배열의 크기는 리터럴 상수, 매크로 상수로 지정할 수 있다.

```
#define MAX 5  
int arr[MAX];        // 배열의 크기를 지정할 때 매크로 상수를 사용할 수 있다.
```

배열_배열의 기본

❖ 배열의 선언(3/3)

- 배열 이름으로부터 배열의 크기(원소의 개수)를 구할 수 있다.

```
int arr[5];
int size1, size2, size3;
size1 = sizeof(arr) / sizeof(arr[0]);    // 배열의 크기(원소의 개수)
printf("배열의 크기: %d\n", size1);

size2 = sizeof(arr) / sizeof(arr[1]);    // 배열의 크기
size3 = sizeof(arr) / sizeof(int);      // 배열의 크기
```

배열_배열의 기본

❖ 배열의 크기를 구해서 사용하는 이유

배열의 크기로 리터럴 상수를
직접 사용하는 경우

```
10
int arr[5];
int sum = 0;
int i;

for (i = 0; i < 5; i++)
{
    scanf("%d", &arr[i]);
    sum += arr[i];
}
printf("sum = %d\n", sum);
```

배열의 크기를 변경하면
소스 나머지부분도 모두
수정해야 한다.

배열의 크기를 변수에
구해서 사용하는 경우

```
10
int arr[5];
int sum = 0;
int size =
    sizeof(arr)/sizeof(arr[0]);
int i;

for (i = 0; i < size; i++)
{
    scanf("%d", &arr[i]);
    sum += arr[i];
}
printf("sum = %d\n", sum);
```

배열의 크기를 변경하
려면 배열의 선언문만
수정하면 된다.

size는 그대로
사용할 수 있다.

배열_배열의 기본

❖ 배열의 크기로 매크로 상수를 사용하는 경우

```
#define ARR_SIZE 10  
  
int arr[ARR_SIZE];  
int sum = 0;  
  
int i;  
  
for (i = 0; i < ARR_SIZE; i++)  
{  
    scanf("%d", &arr[i]);  
    sum += arr[i];  
}  
printf("sum = %d\n", sum);
```

배열의 크기를 변경하려면 매크로 정의만 수정하면 된다.

배열의 크기를 변경해도 나머지 코드는 수정할 필요가 없다.

배열_배열의 사용

❖ 배열의 선언 및 사용 예

```
01: /* Ex07_01.c */  
02: #include <stdio.h>  
03:  
04: int main(void)  
05: {  
06:     int arr[5]; ← int 배열의 선언  
07:     int i;  
08:  
09:     arr[0] = 10; ← 배열의 사용  
10:     arr[1] = 20;  
11:     arr[2] = 30;  
12:     arr[3] = 40;  
13:     arr[4] = 50; ←  
14:  
15:     for(i = 0 ; i < 5 ; i++)  
16:         printf("arr[%d] = %d\n", i, arr[i]); ← 배열의 각 원소 출력  
17:  
18:     return 0;  
19: }
```

실행 결과

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

배열_배열의 초기화

❖ 배열의 초기화(1/4)

- { } 안에 배열 원소의 초기값을 콤마(,)로 나열한다.
- 배열의 0번째 원소부터 배열의 초기값이 나열된 순서대로 초기화한다.
- 배열도 초기화를 하지 않으면 쓰레기 값을 갖는다.
- 배열도 초기화해서 사용하는 것이 좋다.

초기화되지 않은 경우

```
int arr [5];
```

초기화되지 않은 배열은
쓰레기 값을 갖습니다.

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

배열 원소의
초기값

배열의 원소들을
순서대로 초기화



arr[0] arr[1] arr[2] arr[3] arr[4]

배열_배열의 초기화

❖ 가장 기본적인 배열의 초기화 예

```
03 int main(void)
04 {
05     int arr[5] = { 1, 2, 3, 4, 5 };      // 배열의 크기만큼 초기값을 지정한다.
06     int i;
07
08     printf("arr = ");
09     for (i = 0; i < 5; i++)
10         printf("%d ", arr[i]);
11     printf("\n");
12
13     return 0;
14 }
```

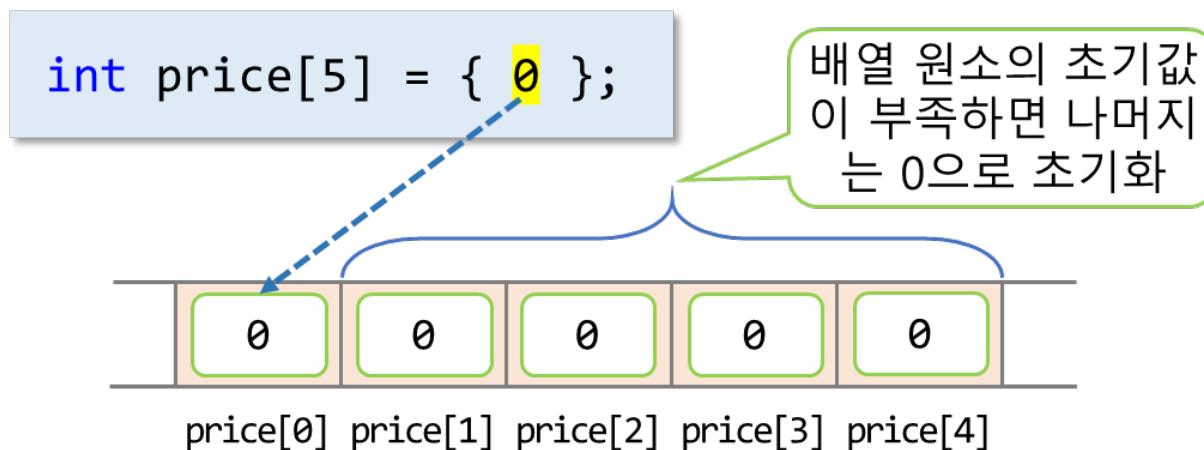
실행결과

arr = 1 2 3 4 5

배열_배열의 초기화

❖ 배열의 초기화(2/4)

- 초기값이 부족하면 나머지 원소는 0으로 초기화한다.



배열_배열의 초기화

❖ 배열의 크기보다 초기값을 적게 지정하는 경우

```
03 int main(void)
04 {
05     int amount[5] = { 1, 1, 5 };           // 1, 1, 5, 0, 0으로 초기화
06     int price[5] = { 0 };                 // 배열 전체를 0으로 초기화
07     int i;
08
09     printf("amount = ");
10    for (i = 0; i < 5; i++)
11        printf("%d ", amount[i]);
12    printf("\n");
13
14    printf("price = ");
15    for (i = 0; i < 5; i++)
16        printf("%d ", price[i]);
17    printf("\n");
18    return 0;
19 }
```

실행결과

```
amount = 1 1 5 0 0
price = 0 0 0 0 0
```

배열_배열의 초기화

❖ 배열의 초기화(3/4)

- 초기값을 원소의 개수보다 많으면 컴파일 에러가 발생한다.

```
int amount[5] = { 1, 1, 5, 2, 10, 3 };
```

- { } 안을 비워 두면 컴파일 에러가 발생한다.

```
int amount[5] = { };
```

- 초기값을 지정하지 않고 배열의 크기를 생략하면 컴파일 에러가 발생한다.

```
int scores[];
```

배열_배열의 초기화

❖ 배열의 초기화(4/4)

- 배열의 초기값을 지정할 때는 배열의 크기를 생략할 수 있다.

```
int arr[ ] = { 1, 2, 3 };
```

초기값이 3개이므로 크기가 3인 배열로 선언된다.

```
int arr[ ];
```

초기값 없이 배열의 크기를 생략했으므로 컴파일 에러

초기값의 개수로
배열의 크기를
유추할 수 있다.

4개

```
int scores[ ] = { 97, 72, 56, 88 };
```

97

72

56

88

scores[0] scores[1] scores[2] scores[3]

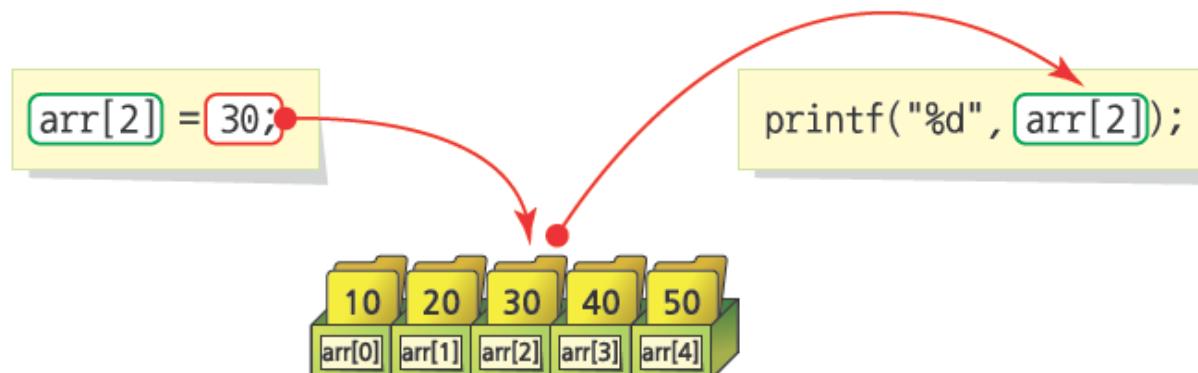
배열_배열의 사용

❖ 배열 원소의 사용(1/2)

- 배열의 각 원소에 접근하려면 첨자 또는 인덱스(index)를 이용한다.

```
arr[0] = 5;           // 배열의 원소에 값을 대입할 수 있다.  
arr[1] = arr[0] + 10; // 배열의 원소를 수식에 이용할 수 있다.  
arr[2] = add(arr[0], arr[1]); // 배열의 원소를 함수의 인자로 전달할 수 있다.  
printf("정수를 2개 입력하세요: ");  
scanf("%d %d", &arr[3], &arr[4]); // 배열의 원소에 정수 값을 입력받을 수 있다.
```

- 배열의 인덱스는 항상 0부터 시작하고, 배열 크기보다 1 작은 값까
지 사용할 수 있다.



배열_배열의 사용

❖ 배열 원소의 사용(2/2)

- 배열은 주로 for문과 함께 사용된다.

```
for (i = 0; i < ARR_SIZE; i++)
    printf("%d ", arr[i]);      // i번째 원소를 출력한다.
```

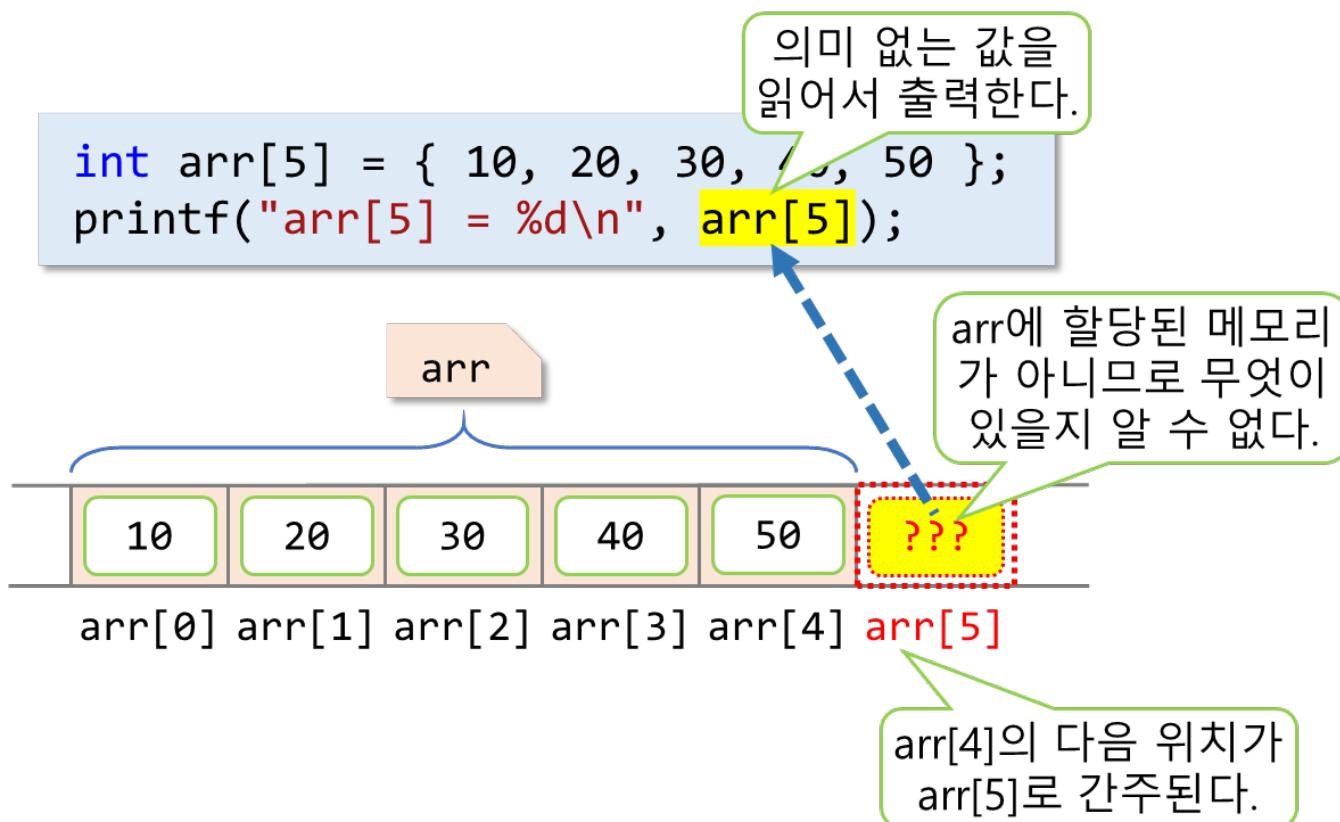
- 배열의 인덱스에는 변수나 변수를 포함한 수식을 사용할 수 있다.

```
arr[i] = arr[i-1] * 2;          // 배열의 인덱스로 정수식을 사용할 수 있다.
```

배열_배열의 사용

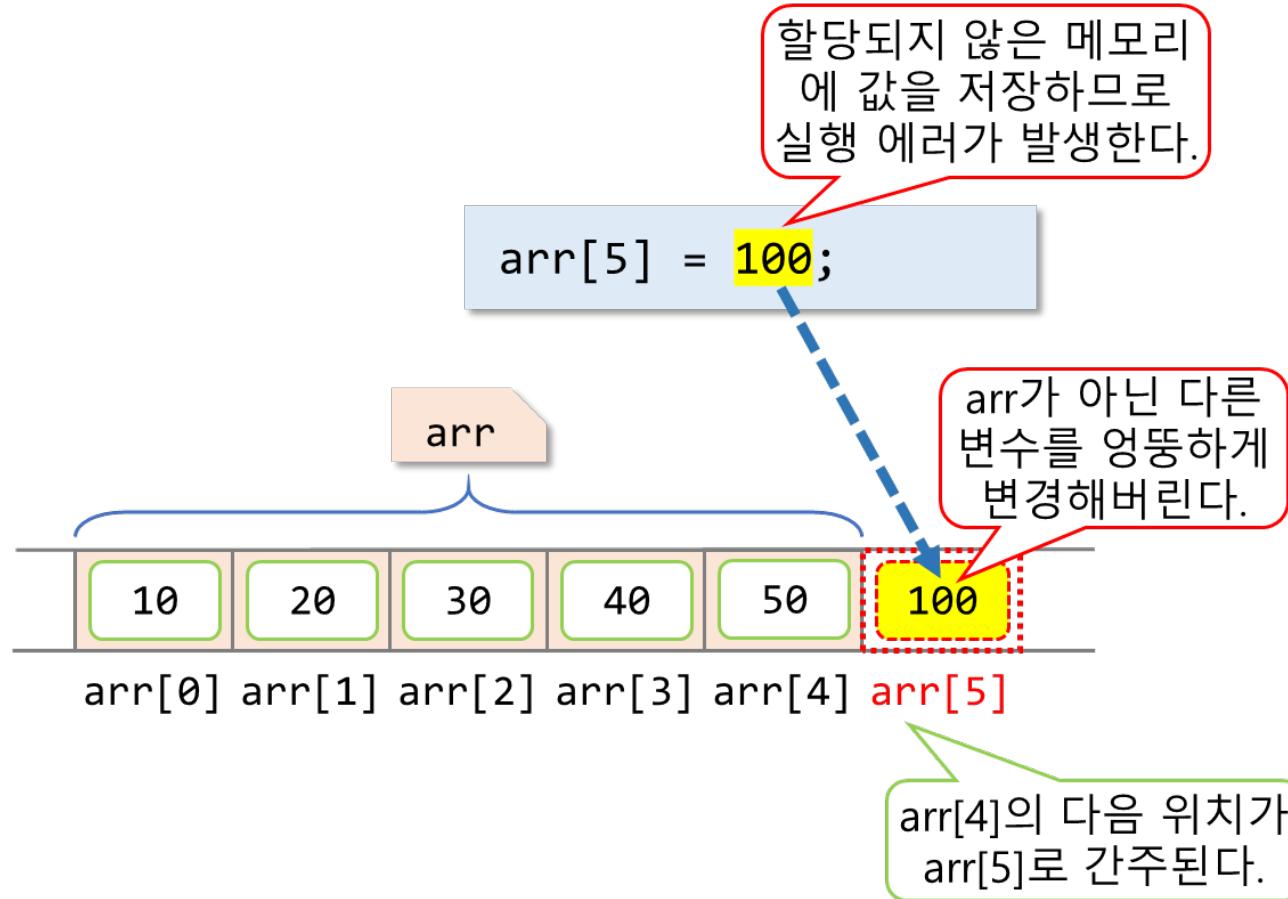
❖ 배열 인덱스의 유효 범위(1/2)

- 배열 인덱스의 유효 범위는 **0 ~ (배열의 크기 - 1)** 사이의 값이다.
- 잘못된 인덱스를 사용하면 실행 에러가 발생할 수 있다.



배열_배열의 사용

❖ 배열 인덱스의 유효 범위(2/2)



배열_배열의 사용

❖ 잘못된 인덱스의 사용 예

```
03 int main(void)
04 {
05     int arr[5] = { 10, 20, 30, 40, 50 };
06     int i;
07
08     printf("arr = ");
09     for (i = 0; i < 5; i++)
10         printf("%d ", arr[i]);
11     printf("\n");
12
13     printf("arr[5] = %d\n", arr[5]);      // 할당되지 않은 메모리를 읽어 온다.
14     arr[5] = 100;                      // 할당되지 않은 메모리를 변경한다. (실행 에러)
15
16     return 0;
17 }
```

실행결과

arr = 10 20 30 40 50

arr[5] = -858993460

쓰레기 값

배열_배열의 활용

❖ 정수형 배열의 합계와 평균 구하기(1/2)

```
01: /* Ex07_04.c */
02: #include <stdio.h>
03:
04: #define MAX 5 ← 배열의 크기로 사용될 매크로 상수
05:
06: int main(void)
07: {
08:     int arr[MAX] = { 0 }; ← int형 배열의 선언 및 초기화
09:     int i;
10:     int sum;
11:     double average;
12:
13:     printf("%d개의 정수를 입력하세요 : ", MAX);
14:     for(i = 0 ; i < MAX ; i++)
15:         scanf("%d", &arr[i]); ← 배열의 입력
16:
17:     for(i = 0, sum = 0 ; i < MAX ; i++)
18:         sum += arr[i]; ← 배열의 합계 구하기
19:
```

배열_배열의 활용

❖ 정수형 배열의 합계와 평균 구하기(2/2)

```
20:     average = (double)sum / (double)MAX; ←———— 배열의 평균 구하기
21:     printf("합계 : %d, 평균 : %5.2f\n", sum, average);
22:
23:     return 0;
24: }
```

실행 결과



5개의 정수를 입력하세요 : 283 456 232 121 34
합계 : 1126, 평균 : 225.20

배열_배열의 복사

- ❖ 데이터형과 크기가 같은 경우에도 배열에 다른 배열을 대입 할 수 없다.

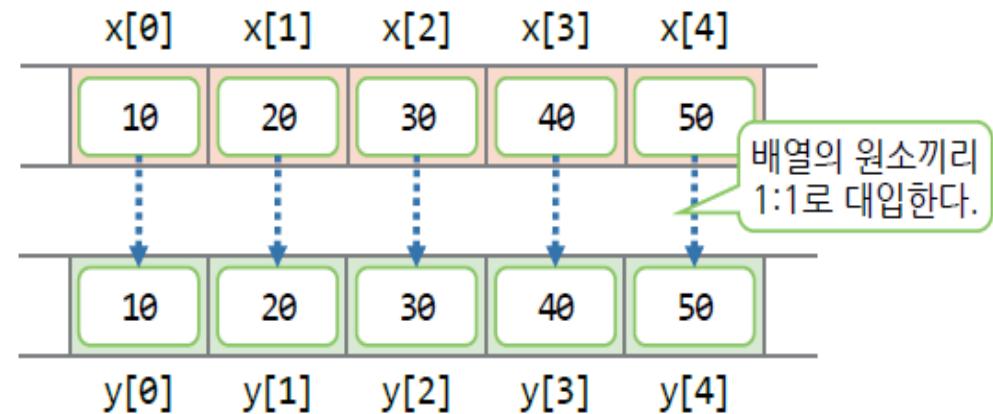
```
int x[ARR_SIZE] = { 10, 20, 30, 40, 50 };
int y[ARR_SIZE] = { 0 };

🚫 y = x;
```

배열 이름으로 배열 전체를 대입할 수 없다.

- ❖ 배열을 복사하려면 배열의 모든 원소에 대하여 원소끼리 대입한다.

```
for (i = 0; i < ARR_SIZE; i++)
    y[i] = x[i];
```



배열_배열의 복사

```
01 #include <stdio.h>
02 #define ARR_SIZE 5
03
04 int main(void)
05 {
06     int x[ARR_SIZE] = { 10, 20, 30, 40, 50 };
07     int y[ARR_SIZE] = { 0 };
08     int i;
09
10    for (i = 0; i < ARR_SIZE; i++)
11        y[i] = x[i];
12
13    for (i = 0; i < ARR_SIZE; i++)
14        printf("%d ", y[i]);
15    printf("\n");
16 }
```

실행 결과

10 20 30 40 50

x와 y는 데이터형과 크기가 같은 배열이다.

배열의 원소끼리 1:1로 대입한다.

배열_배열의 비교

- ❖ 두 배열이 같은지 비교하기 위해서 == 연산자로 직접 배열을 비교하면 배열의 시작 주소를 비교한다.
 - 배열의 이름은 배열의 시작 주소이기 때문이다.

x와 y의 주소가
같은지 비교한다.

```
int x[ARR_SIZE] = { 10, 20, 30, 40, 50 };
int y[ARR_SIZE] = { 10, 20, 30, 40, 50 };
if (x == y)
    printf("두 배열의 주소가 같습니다.\n");
```

- ❖ 배열의 내용이 같은지 비교하려면 for문을 이용해서 원소끼리 비교해야 한다.
 - 모든 원소의 값이 같으면, 배열 전체의 내용이 같다.

배열_배열의 비교(1)

```
01 #include <stdio.h>
02 #define ARR_SIZE 5
03
04 int main(void)
05 {
06     int x[ARR_SIZE] = { 10, 20, 30, 40, 50 };
07     int y[ARR_SIZE] = { 10, 20, 30, 40, 50 };
08     int i;
09     int is_equal;
10
11     if (x == y)
12         printf("두 배열의 주소가 같습니다.\n");
13     else
14         printf("두 배열의 주소가 다릅니다.\n");
```

배열이 같은지를
나타내는 변수

x와 y의 주소가
같은지 비교한다.

배열_배열의 비교(2)

```
16 is_equal = 1;  
17 for (i = 0; i < ARR_SIZE; i++) {  
18     if (x[i] != y[i]) {  
19         is_equal = 0;  
20         break;  
21     }  
22 }  
23 if (is_equal == 1)  
    printf("두 배열의 내용이 같습니다.\n");  
25 else  
    printf("두 배열의 내용이 다릅니다.\n");  
27 }
```

배열의 모든 원소가 같은지 비교한다.

서로 다른 원소가 있으면 더 이상 비교할 필요가 없다.

실행 결과

두 배열의 주소가 다릅니다.
두 배열의 내용이 같습니다.

배열_다차원 배열

❖ 다차원 배열

- 배열의 원소에 접근하기 위해서 인덱스를 둘 이상 사용하는 경우를 말한다.
 - 행렬이나 표, 폭과 높이가 있는 이미지 데이터 등을 나타내는 데 이용



점수 3개를 저장하는 배열
이 5개 필요하다. (점수 3개
× 학생 5명)

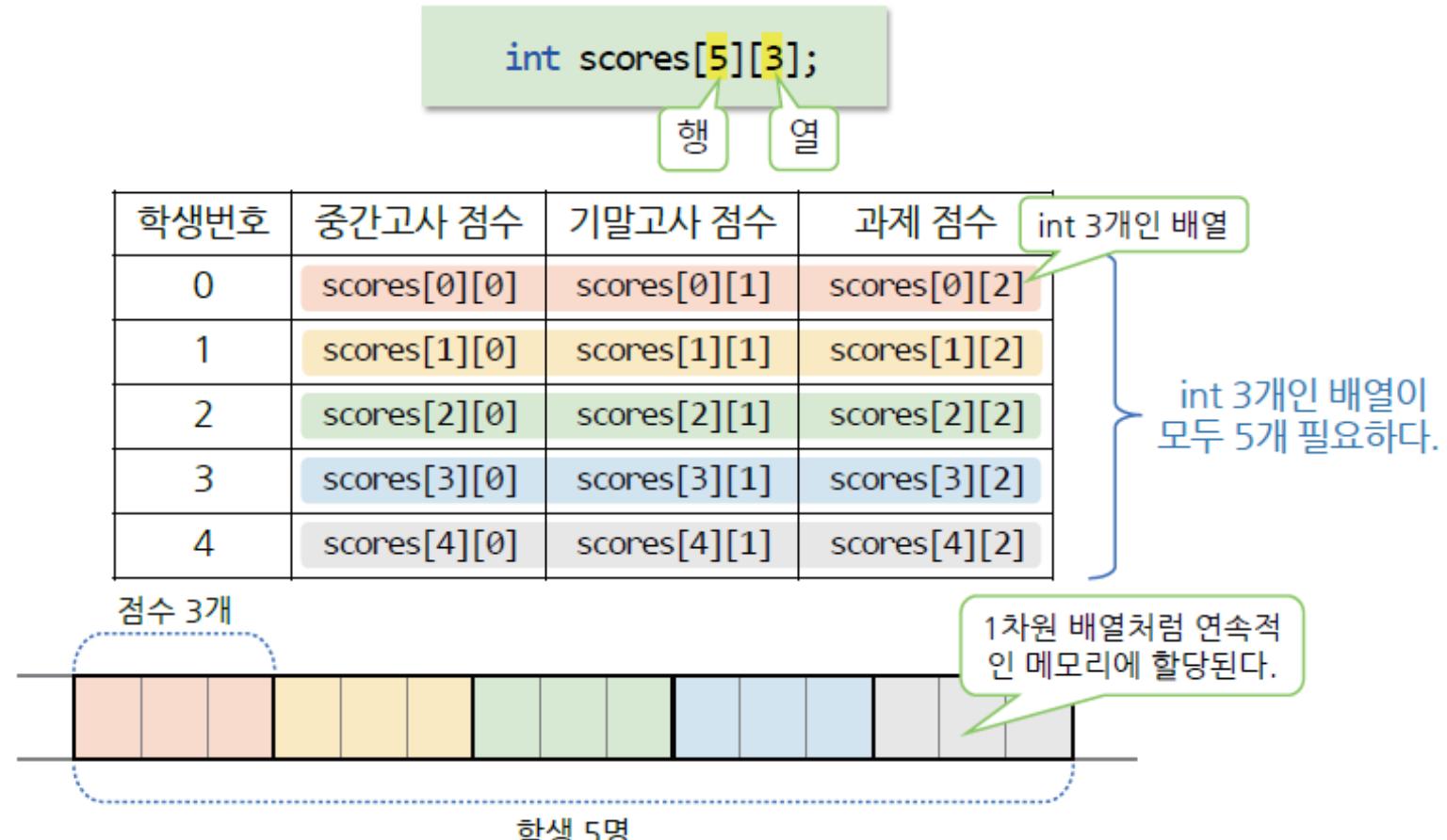
```
int scores[5][3];
```

- 다차원 배열도 일차원 배열처럼 연속된 메모리에 할당된다.
- 다차원 배열의 차수에는 제한이 없다.

배열_다차원 배열

❖ 2차원 배열

- 행(row)과 열(column)의 개념으로 이해할 수 있다.



배열_다차원 배열

❖ 이차원 배열의 선언(1/2)

형식

데이터형 배열명[행크기][열크기];

사용예

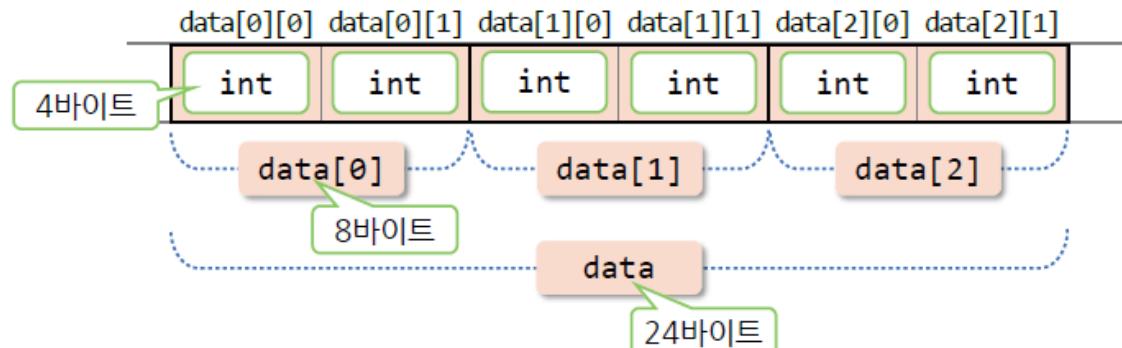
```
int scores[5][3];
double matrix[4][4];
char passwords[10][32];
```

원소의 개수는

'(열 크기)×(행 크기)'개

```
#define ROW 3
#define COL 2
int data[ROW][COL];
```

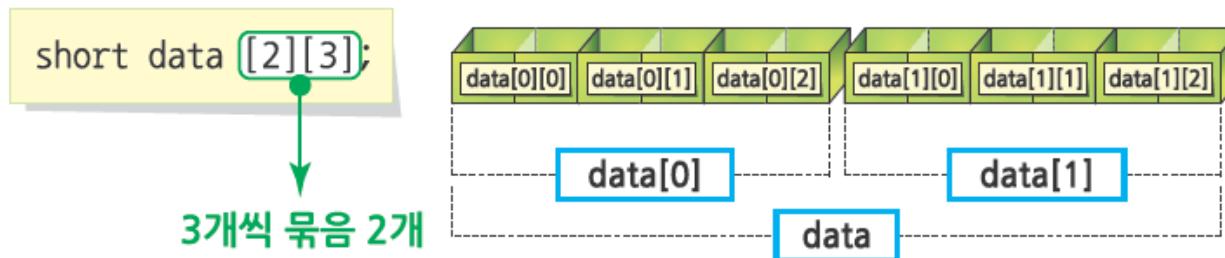
배열의 크기를 매크로
상수로 지정한다.



배열_다차원 배열

❖ 이차원 배열의 선언(2/2)

- 2차원 배열의 원소들도 1차원 배열처럼 메모리에 연속적으로 할당된다.



```
printf("sizeof(data)
```

```
= %d\n", sizeof(data));
```

배열 전체의 바
이트 크기

```
printf("sizeof(data[0])
```

```
= %d\n", sizeof(data[0]));
```

int 2개인 data[0]의
바이트 크기

```
printf("sizeof(data[0][0])
```

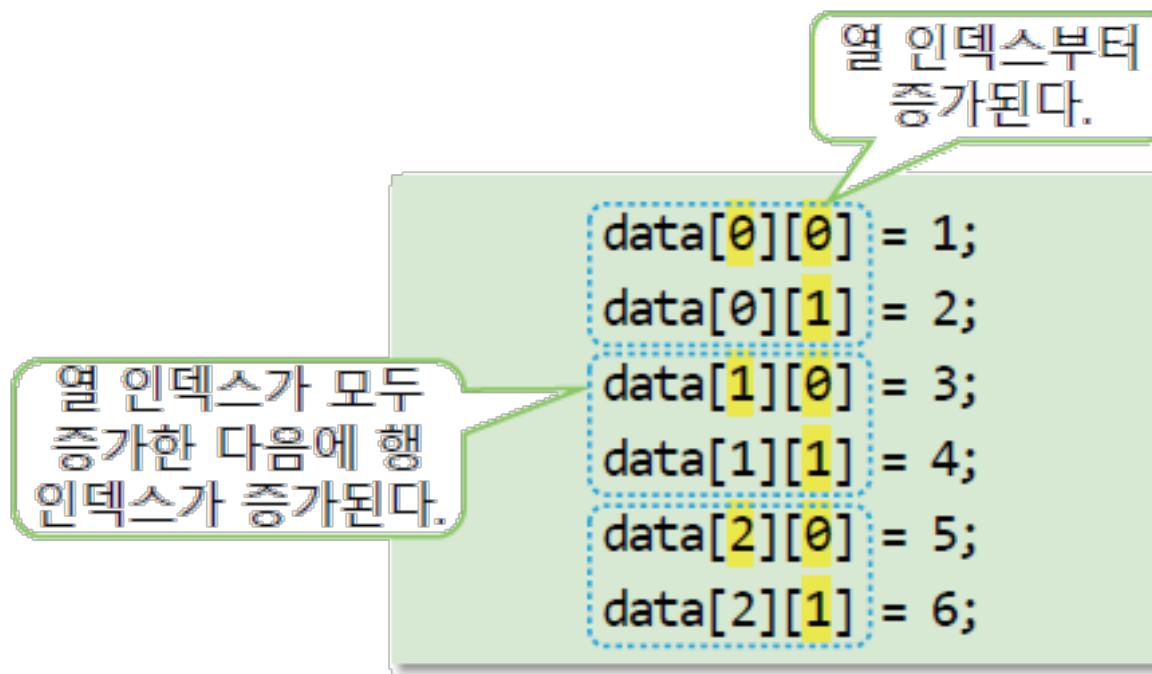
```
= %d\n", sizeof(data[0][0]));
```

int인 data[0][0]의
바이트 크기

배열_다차원 배열

❖ 이차원 배열의 사용(1/2)

- 이차원 배열의 원소에 접근할 때 **인덱스를 두 개** 사용한다.
- 2차원 배열에서는 열 인덱스가 먼저 증가되고, 그 다음에 행 인덱스가 증가된다.
- 인덱스의 개수가 배열의 차수를 결정한다.



```
data[0][0] = 1;
data[0][1] = 2;
data[1][0] = 3;
data[1][1] = 4;
data[2][0] = 5;
data[2][1] = 6;
```

열 인덱스가 모두 증가한 다음에 행 인덱스가 증가된다.

열 인덱스부터 증가된다.

배열_다차원 배열

❖ 이차원 배열의 사용(2/2)

- 2차원 배열은 중첩된 for와 함께 사용할 수 있다.
 - 안쪽 for문을 이용해서 열 인덱스를 증가시키고, 바깥쪽 for문을 이용해서 행 인덱스를 증가시키면 2차원 배열의 원소가 메모리에 할당된 순서대로 접근할 수 있다.

```
for (i = 0, k = 0; i < ROW; i++)  
    for (j = 0; j < COL; j++)  
        data[i][j] = ++k;
```

행 인덱스를 증가시킨다.

열 인덱스를 증가시킨다.

2차원 배열의 원소가 메모리에
할당된 순서대로 접근한다.

배열_다차원 배열

❖ 이차원 배열의 선언 및 사용 예(1/2)

```
01 #include <stdio.h>
02 #define ROW 3
03 #define COL 2
04
05 int main(void)
06 {
07     int data[ROW][COL];
08     int i, j, k;
09
10    for (i = 0, k = 0; i < ROW; i++)
11        for (j = 0; j < COL; j++)
12            data[i][j] = ++k;
13}
```

```
data[0][0] = 1;
data[0][1] = 2;
data[0][2] = 3;
data[1][0] = 4;
data[1][1] = 5;
data[1][2] = 6;
```

배열원소 직접 할당

이차원 배열의 값 할당

int[2]를 3개 메모리에 할당 한다.

행 인덱스를 증가시킨다.

data 배열의 각 원소가 할당된 순서대로 접근한다.

열 인덱스를 증가시킨다.

배열_다차원 배열

❖ 이차원 배열의 선언 및 사용 예(2/2)

```
14     for (i = 0; i < ROW; i++) {  
15         for (j = 0; j < COL; j++)  
16             printf("%3d ", data[i][j]);  
17         printf("\n");  
18     }  
19  
20     printf("sizeof(data)      = %d\n", sizeof(data));  
21     printf("sizeof(data[0])    = %d\n", sizeof(data[0]));  
22     printf("sizeof(data[0][0]) = %d\n", sizeof(data[0][0]));  
23 }
```

실행 결과

1 2

3 4

5 6

sizeof(data) = 24

sizeof(data[0]) = 8

sizeof(data[0][0]) = 4

배열_다차원 배열

❖ 이차원 배열의 초기화(1/4)

- 일차원 배열처럼 {}를 이용한다. 초기값을 열 크기의 개수만큼씩 {}로 묶어서 다시 {} 안에 나열한다.

```
int data[3][2] = {  
    {10, 20}, {30, 40}, {50, 60},  
};
```

data[0]을 초기화 data[1]을 초기화 data[2]을 초기화

- 일차원 배열처럼 {} 안에 초기값만 나열할 수도 있다.
 - 배열의 원소가 메모리에 할당된 순서대로 초기화한다.

```
int data[3][2] = { 10, 20, 30, 40, 50, 60 };
```

data[0][0]부터 할당된 순서대로 초기화

배열_다차원 배열

❖ 이차원 배열의 초기화(2/4)

- 초기값을 생략하면 나머지 원소를 0으로 초기화한다.

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

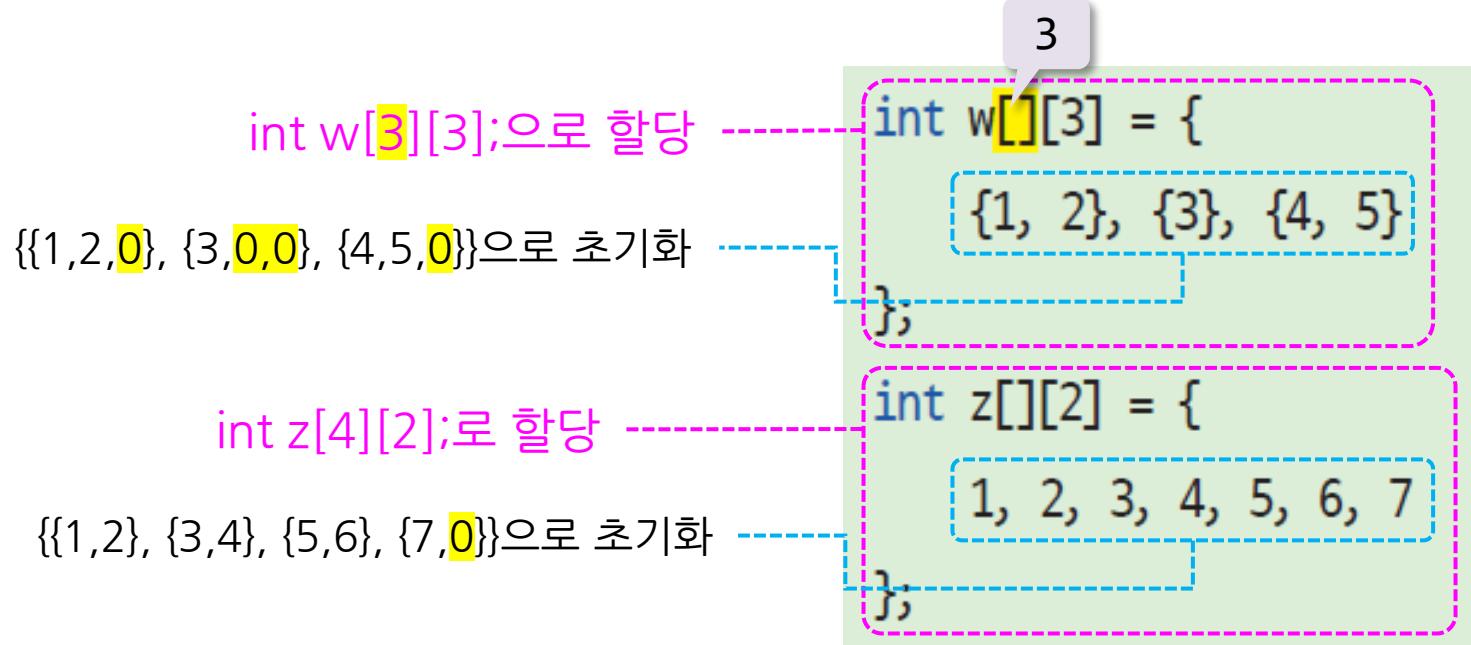
→ {{1,2,3}, {4,5,0}, {6,0,0}, {0,0,0}}으로 초기화

→ {{1, 2}, {3, 0}, {0, 0}}으로 초기화

배열_다차원 배열

❖ 이차원 배열의 초기화(3/4)

- 2차원 배열을 초기화할 때는, 배열의 행 크기를 생략할 수 있다.



배열_다차원 배열

❖ 이차원 배열의 초기화(4/4)

- 이차원 배열의 제1크기만 생략할 수 있고 제2크기는 생략할 수 없다

```
int a[ ][2] = { 1, 2, 3, 4, 5, 6 }; → int a[3][2]로 할당된다.  
int b[3][] = { 1, 2, 3, 4, 5, 6 }; → 컴파일 에러
```

열 크기를 유추할 수 없으므로
컴파일 에러

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

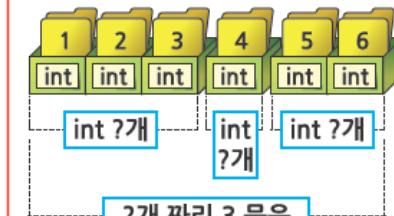
초기값으로부터 2개짜리가
3묶음이라는 것을 알 수 있습니다.



🚫
int v[3][] = {
 {1, 2}, {3}, {4, 5, 6}
};

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

몇 개씩 묶을지 알 수 없습니다.



배열_다차원 배열

❖ 이차원 배열의 초기화 예

```
02 #define ROW 3
03 #define COL 2
04
05 int main(void)
06 {
07     int data[ROW][COL] = {
08         {10, 20}, {30, 40}, {50, 60},
09     };
10     int i, j;
11
12     for (i = 0; i < ROW; i++) {
13         for (j = 0; j < COL; j++)
14             printf("%3d ", data[i][j]);
15         printf("\n");
16     }
17     return 0;
18 }
```

실행결과

```
10 20
30 40
50 60
```

배열의 활용_함수로 인자로 배열 전달하기

❖ 배열을 매개변수로 갖는 함수의 정의

- 함수의 매개변수로 배열을 선언할 때, 배열의 크기는 생략한다.
- 함수 안에서 배열의 크기가 필요하면 배열의 크기도 매개변수로 받아온다.

크기를 지정하지 않고
배열로 선언한다.

배열의 크기도 매개
변수로 받아온다.

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

함수 안에서 배열의 크기
가 필요하면 매개변수로
받아온 값을 이용한다.

배열의 활용_함수로 인자로 배열 전달하기

❖ 배열을 매개변수로 갖는 함수의 정의 방법

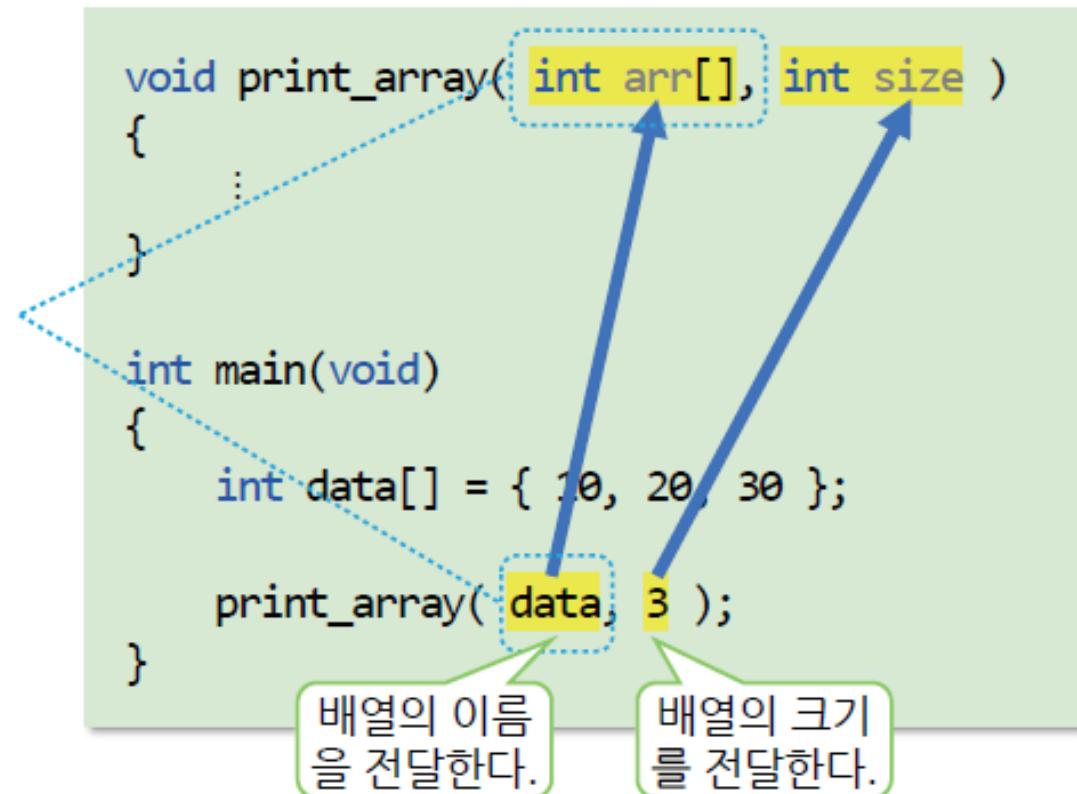
- ① 함수의 매개변수로 배열을 선언할 때는 배열의 원소형, 매개변수명(배열명)과 []를 적어준다. [] 안에 배열의 크기를 쓰지 않고 비워둔다.
- ② 배열의 크기를 전달받기 위한 정수형의 매개변수가 필요하다.
- ③ 함수 안에서 배열의 크기가 필요할 때는 매개변수로 전달받은 배열의 크기를 이용한다.

배열의 활용_함수로 인자로 배열 전달하기

❖ 배열을 매개변수로 갖는 함수의 호출(1/2)

- 배열의 이름과 배열의 크기를 인자로 전달한다.
 - [] 없이 배열의 이름만 써준다.

배열의 원소형이
같아야 한다.



배열의 활용_함수로 인자로 배열 전달하기

❖ 배열을 매개변수로 갖는 함수의 호출(2/2)

- 함수 안에서는 항상 매개변수로 전달받은 배열의 크기를 이용한다.

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

```
print_array(x, 3);
```

x가 크기가 3인 배열인 것처럼 {1, 2, 3}만 출력한다.

- 매개변수의 원소형과 인자로 전달하는 배열의 원소형이 같아야 한다.

```
float grades[3] = { 4.0, 4.3, 3.7 };
```

🚫 `print_array(grades, 3);`

인자와 매개변수의 원소형
이 같아야 한다.

배열의 활용_함수로 인자로 배열 전달하기

```
01 #include <stdio.h>
02 void print_array(int arr[], int size); ----- 함수 선언
```

```
03
```

```
04 int main(void)
```

```
05 {
```

```
06     int data[3] = { 10, 20, 30 };
```

```
07     int x[] = { 1, 2, 3, 4, 5 };
```

```
08     int size = sizeof(x) / sizeof(x[0]); ----- x 배열의 크기
```

```
09
```

```
10     printf("data = ");
```

```
11     print_array(data, 3);
```

```
12
```

```
13     printf("x      = ");
```

```
14     print_array(x, size);
```

배열 이름과 크기를
인자로 전달한다.

배열의 활용_함수로 인자로 배열 전달하기

```
16     printf("x      = ");
17     print_array(x, 3);
18 }
19
20 void print_array(int arr[], int size)
21 {
22     int i;
23     for (i = 0; i < size; i++)
24         printf("%d ", arr[i]);
25     printf("\n");
26 }
```

x 배열을 크기가 3인 배열처럼 출력한다.

배열형의 매개변수에서 크기는 생략한다.

배열의 크기는 별도의
매개변수로 받아와야 한다.

실행 결과

```
data = 10 20 30
x      = 1 2 3 4 5
x      = 1 2 3
```

배열의 활용_ 배열의 탐색과 정렬

❖ 탐색(search) 또는 검색

- 주어진 데이터 집합에서 조건이 만족하는 데이터를 찾는 것
- 검색할 상품명을 입력하면 여러 쇼핑몰의 상품 정보 중에서 상품명이 일치하는 항목을 찾아서 화면에 표시하는 기능

❖ 배열의 탐색

- 탐색 키(key)와 같은 값을 가진 원소를 찾는다.
- 순차 탐색(sequential search)
 - 배열의 0번째 원소부터 순서대로 탐색 키와 비교해서 값이 같은 원소를 찾는다.

배열의 활용_ 배열의 탐색과 정렬

```
01 #define _CRT_SECURE_NO_WARNINGS
02 #include <stdio.h>
03
04 void print_array(int arr[], int size)
05 {
06     int i;
07     for (i = 0; i < size; i++)
08         printf("%d ", arr[i]);
09     printf("\n");
10 }
11
12 int main(void)
13 {
14     int data[] = { 12, 34, 51, 22, 91, 12, 15 };
```

크기를 생략한
배열을 선언한다.

배열의 활용_ 배열의 탐색과 정렬

```
15 int size, i;  
16 int key;  
17  
18 size = sizeof(data) / sizeof(data[0]);  
19 printf("data = ");  
20 print_array(data, size);  
21  
22 printf("찾을 값(키)? ");  
23 scanf("%d", &key);  
24 for (i = 0; i < size; i++) {  
25     if (data[i] == key)  
26         printf("찾은 원소의 인덱스: %d\n", i);  
27 }  
28 }
```

실행 결과

```
data = 12 34 51 22 91 12 15  
찾을 값(키)? 12  
찾은 항목의 인덱스 : 0  
찾은 항목의 인덱스 : 5
```

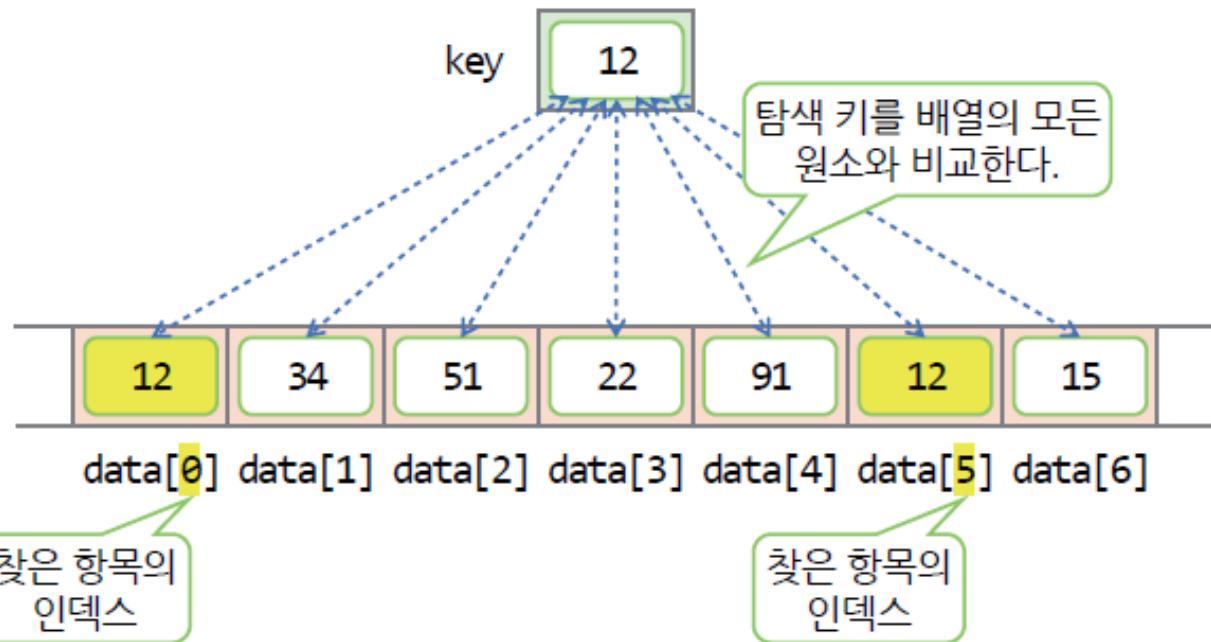
탐색 키를 입력받는다.

탐색 키와 값이 같은 원소를 찾는다.

배열의 활용_배열의 탐색과 정렬

❖ 예제7-10의 실행과정

- 탐색 키와 값이 같은 원소를 모두 찾아서 인덱스를 출력한다.



- 탐색 키와 일치하는 첫 번째 원소만 찾고 탐색을 종료하려면 `break`로 `for`문을 탈출한다.

배열의 활용_ 배열의 탐색과 정렬

❖ 정렬(sort)

- 주어진 데이터 항목을 지정된 순서로 나열하는 것
- 가격 비교 사이트에서 검색 결과로 표시된 상품 목록을 낮은 가격 순으로 보거나 높은 가격 순으로 확인하는 기능

❖ 배열의 정렬

- 원소들을 비교해서 크기가 커지는 순서 또는 작아지는 순서로 나열 한다.
 - 오름차순(ascending order) : 크기가 커지는 순서
 - 내림차순(descending order) : 크기가 작아지는 순서
- 선택 정렬(selection sort)
 - 전체 배열의 원소 중 가장 작은 값을 선택해서 배열의 0번 원소로 옮기고, 그 다음 작은 값을 선택해서 배열의 1번 원소로 옮기는 식으로 진행된다.

배열의 활용_ 배열의 탐색과 정렬

❖ 오름차순 선택정렬(1/2)

```
01 #include <stdio.h>
02 #define SIZE 5
04 void print_array(int arr[], int size)
05 {
06     int i;
07     for (i = 0; i < size; i++)
08         printf("%d ", arr[i]);
09     printf("\n");
10 }
12 int main(void)
13 {
14     int data[SIZE] = { 52, 31, 28, 17, 46 };
15     int i, j, temp;
16     int index_min;
```

실행 결과

i = 0 일때 정렬 결과 : 17 31 28 52 46
i = 1 일때 정렬 결과 : 17 28 31 52 46
i = 2 일때 정렬 결과 : 17 28 31 52 46
i = 3 일때 정렬 결과 : 17 28 31 46 52

아직 정렬되지 않은
원소 중 가장 작은 원
소의 인덱스

배열의 활용_배열의 탐색과 정렬

❖ 오름차순 선택정렬(2/2)

```
18     for (i = 0; i < SIZE - 1; i++) {  
19         index_min = i;  
20         for (j = i + 1; j < SIZE; j++) {  
21             if (data[index_min] > data[j])  
22                 index_min = j;  
23         }  
24         if (i != index_min) {  
25             temp = data[i];  
26             data[i] = data[index_min];  
27             data[index_min] = temp;  
28         }  
29     }  
30     printf("i = %d 일때 정렬 결과 : ", i);  
31     print_array(data, SIZE);  
32 }  
33 }  
34 }
```

data[0]~data[i-1]는 정렬된 상태

data[i]~data[SIZE-1] 중에서 가장 작은 원소의 인덱스를 찾는다.

data[i]를
data[index_min]
와 맞바꾼다.

배열의 활용_배열의 탐색과 정렬

❖ 선택 정렬의 수행과정



`data[0]`과 `data[1]~data[4]`를 비교해서 `data[0]`을 가장 작은 `data[3]`과 맞바꾼다.

`data[1]`과 `data[2]~data[4]`를 비교해서 `data[1]`을 가장 작은 `data[2]`와 맞바꾼다.

`data[2]`와 `data[3]~data[4]`를 비교해서 `data[2]`가 가장 작으므로 그대로 둔다.

`data[3]`과 `data[4]`를 비교해서 `data[3]`을 더 작은 `data[4]`와 바꾼다.

예제 1.

- ❖ 크기가 10인 정수형 배열을 선언하고 초기화 한 후, 배열 원소 중 홀수만 출력하는 프로그램을 작성하시오.

```
C:\Windows\system32\cmd.exe
배열 원소값을 입력하시오. 1 2 3 4 5 6 7 8 9 10
arr[1]=1
arr[3]=3
arr[5]=5
arr[7]=7
arr[9]=9
계속하려면 아무 키나 누르십시오 . . .
```

예제 2.

- ❖ 크기가 10인 정수형 배열을 선언하고, 2번 배열원소까지만 값을 할당하고, 나머지는 0으로 초기화한다. 배열 원소값이 0인 경우에만 값을 할당받는 프로그램을 작성하시오.

```
C:\WINDOWS\system32\cmd.exe
4번째 배열원소값을 입력하시오.5
5번째 배열원소값을 입력하시오.3
6번째 배열원소값을 입력하시오.6
7번째 배열원소값을 입력하시오.2
8번째 배열원소값을 입력하시오.7
9번째 배열원소값을 입력하시오.4
10번째 배열원소값을 입력하시오.9
==배열 원소 출력==
arr[0]=1
arr[1]=2
arr[2]=3
arr[3]=5
arr[4]=3
arr[5]=6
arr[6]=2
arr[7]=7
arr[8]=4
arr[9]=9
계속하려면 아무 키나 누르십시오 . . .
```

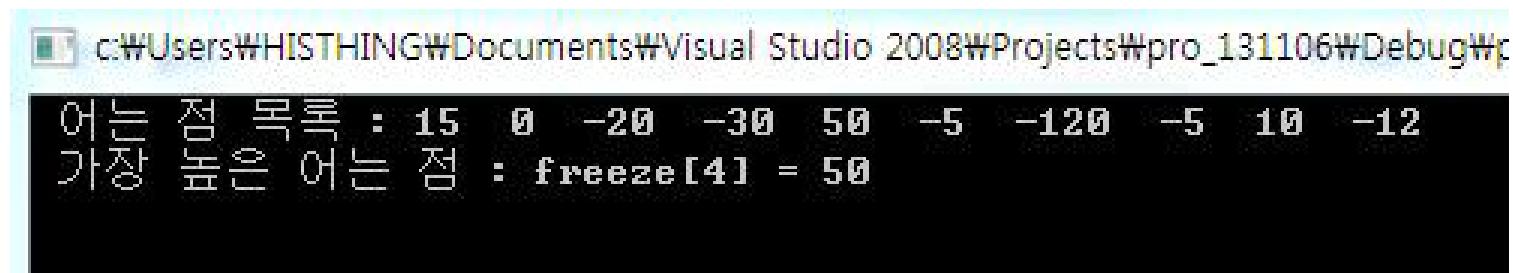
예제 3.

❖ 크기가 10인 정수형 배열을 선언하고, 등차수열로 값을 채우려고 한다. 첫번째 항의 값과 공차를 입력받아서 배열원소 값을 채우고 출력하는 프로그램을 작성하시오.

```
C:\WINDOWS\system32\cmd.exe
첫째항과 공차를 입력하시오. 1 5
arr[0]=1
arr[1]=6
arr[2]=11
arr[3]=16
arr[4]=21
arr[5]=26
arr[6]=31
arr[7]=36
arr[8]=41
arr[9]=46
계속하려면 아무 키나 누르십시오 . . .
```

예제 4

- ❖ 주어진 프로그램을 참고하여, 열 가지 물질의 어는 점 {15, 0, -20, -30, 50, -5, -120, -5, 10, -12}를 freeze 배열에 저장한 후, 가장 높은 어는 점(최대값)을 구하여 다음과 같이 출력하시오.



```
c:\Users\HISTHING\Documents\Visual Studio 2008\Projects\pro_131106\Debug\c
어는 점 목록 : 15  0  -20  -30  50  -5  -120  -5  10  -12
가장 높은 어는 점 : freeze[4] = 50
```

예제 5

❖ 정수형 배열 score 배열에 점수를 입력받아 저장한 후, 입력된 점수와 점수 평균을 출력하는 프로그램을 작성하려고 한다. 입력될 건수는 매크로 상수로 다음과 같이 선언되어 있다.

- `#define ST_SIZE 5`
- -조건 1. 점수가 저장될 배열은 매크로 상수의 크기로 선언된다.
- -조건 2. 점수를 입력할 때 배열에 저장하고 누적한다.
- -조건 3. 입력이 끝나면 입력된 데이터와 평균을 출력한다.

예제 6

- ❖ 5*5 행렬의 원소를 1부터 25까지라고 할 때, 중첩for문으로 행렬의 원소를 2차원 정수형 배열에 저장하고 출력한다.
- ❖ 2차원 정수형 배열에 저장한 원소 중 행렬의 대각 원소만 출력하는 프로그램을 작성하시오.
 - `#define ARR_SIZE 5`
 - -조건 1. 행렬원소가 저장될 배열은 매크로 상수의 크기로 선언된다.
 - -조건 2. 5*5 행렬의 대각원소를 저장하는 1차원 배열을 선언하여 대각원소의 값을 저장한 후, 출력한다.

 Microsoft Visual Studio 디버그 콘솔

```
1  2  3  4  5  
6  7  8  9  10  
11 12 13 14 15  
16 17 18 19 20  
21 22 23 24 25  
대각원소 :  1  7  13 19 25
```

예제 7(report)

- ❖ 2차원 정수형 배열을 선언하고 초기화한 후, A반 학생 5명의 국,영,수 성적을 2차원 배열에 입력받아 각 학생의 평균을 구하여 출력한다.
- ❖ A반의 각 과목의 평균을 계산하여 출력한다.
 - 조건 1 : 점수가 저장될 2차원 배열의 행과 열의 크기는 매크로 상수의 크기로 선언된다.
 - #define R_SIZE 5
 - #define C_SIZE 3
 - 조건 2 : 학생별 평균과 과목별 평균은 1차원 배열을 선언하여 저장한 후, 출력 한다.

```
Microsoft Visual Studio 디버그 콘솔
1번째 학생의 점수를 입력하시오>> 78 89 90
2번째 학생의 점수를 입력하시오>> 90 80 70
3번째 학생의 점수를 입력하시오>> 60 70 80
4번째 학생의 점수를 입력하시오>> 77 86 90
5번째 학생의 점수를 입력하시오>> 90 88 85
=====학생별 과목 점수 출력=====
78 89 90
90 80 70
60 70 80
77 86 90
90 88 85
=====학생별 평균 출력=====
1번째 학생의 평균 : 85.67
2번째 학생의 평균 : 80.00
3번째 학생의 평균 : 70.00
4번째 학생의 평균 : 84.33
5번째 학생의 평균 : 87.67
=====과목별 평균 출력=====
1번째 과목의 평균 : 79.00
2번째 과목의 평균 : 82.60
3번째 과목의 평균 : 83.00
```

학습정리

❖ 배열

- **배열** : 같은 데이터 형의 변수들을 메모리에 연속적으로 할당하고 같은 이름으로 사용하는 기능
- **배열의 선언** : 배열 원소의 데이터 형, 배열 이름, 배열의 크기가 필요하다.
 - 배열의 크기는 상수로만 지정한다.
- **배열의 사용** : 배열의 각 원소에 접근하려면 인덱스를 사용한다.
 - 인덱스는 항상 0~(배열의 크기-1)사이의 값이다.
- **배열의 초기화** : 배열을 초기화하려면 {} 안에 초기값을 나열한다.
 - 배열을 초기화할 때는 배열의 크기를 생략할 수 있다.
- **다차원 배열** : 배열의 차수에는 제한이 없다.
 - 이차원 배열의 선언 : int arr[2][3];처럼 배열의 제1크기와 제2크기를 지정한다.