배열

2020학년도 1학기
강원도립대학교 ICT드론과

목차

배열의 개념

- 같은 데이터형의 변수를 메모리에 연속적으로 할당하고 같은 이름으로 사용하는 기능
- 배열의 **원소(element)**
  - 배열 안에 들어가는 변수 하나하나
  - 개별적인 변수인 것처럼 사용
- **인덱스(index)**
  - 배열의 각 원소를 구분하기 위한 번호
  - 항상 0부터 시작한다.
- 배열의 모든 원소는 항상 연속된 메모리에 할당된다.

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

   return 0;
}
```

크기가 5인 배열을 선언한다.

반복문 안에서 배열의 각 원소에 대하여 같은 코드를 수행한다.

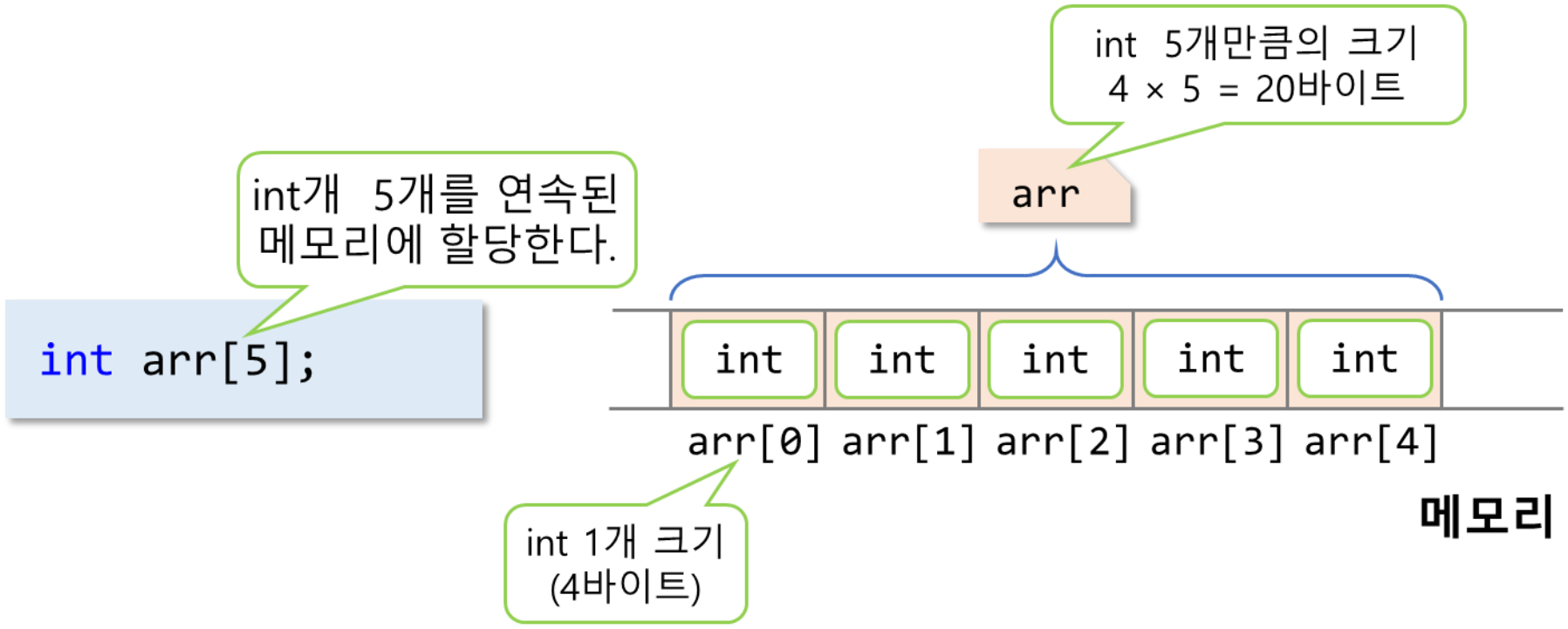
배열의 각 원소는 인덱스로 구분해서 사용한다.

배열의 선언

형식 데이터형 배열명[크기];

사용예

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



배열의 크기 (1/2)

- 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 크기를 지정할 때 매크로 상수를 사용할 수 있다.

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

- const 변수는 변수이므로 배열의 크기를 지정할 때 사용할 수 없다.

```
const int max = 10; // max는 값을 변경할 수 없는 변수이다.
int arr[max];       // 배열의 크기를 지정할 때 변수를 사용할 수 없으므로 컴파일 에러
```

배열의 크기 (2/2)

- 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
```

배열의 크기를 구해서 사용하는 이유

예제 7-1

배열의 크기로 리터럴 상수를 직접 사용하는 경우

```
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);
```

10

10

배열의 크기를 변경하면 소스 나머지부분도 모두 수정해야 한다.

배열의 크기를 변수에 구해서 사용하는 경우

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

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

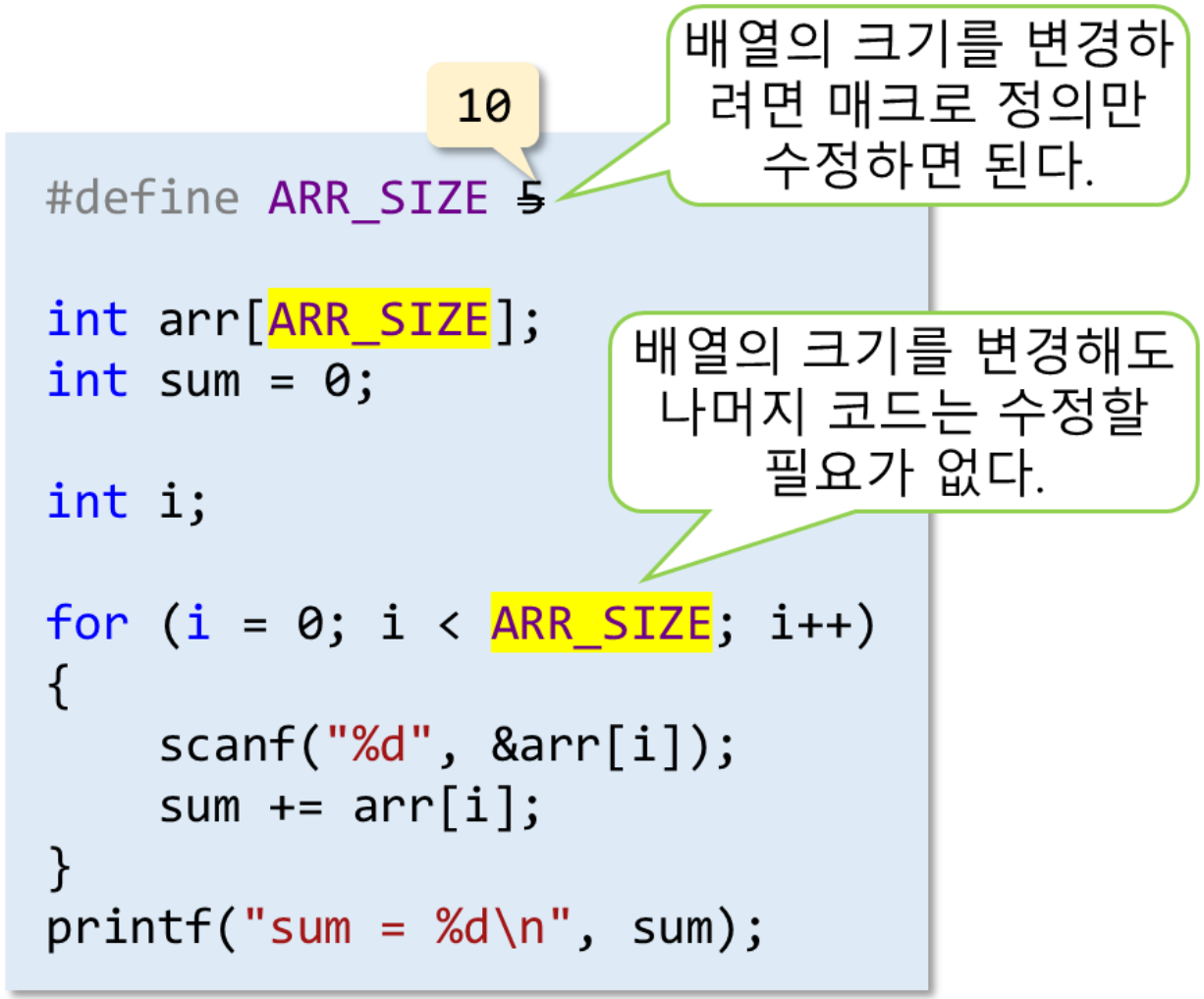
10

배열의 크기를 변경하려면 배열의 선언문만 수정하면 된다.

size는 그대로 사용할 수 있다.

배열의 크기로 매크로 상수를 사용하는 경우

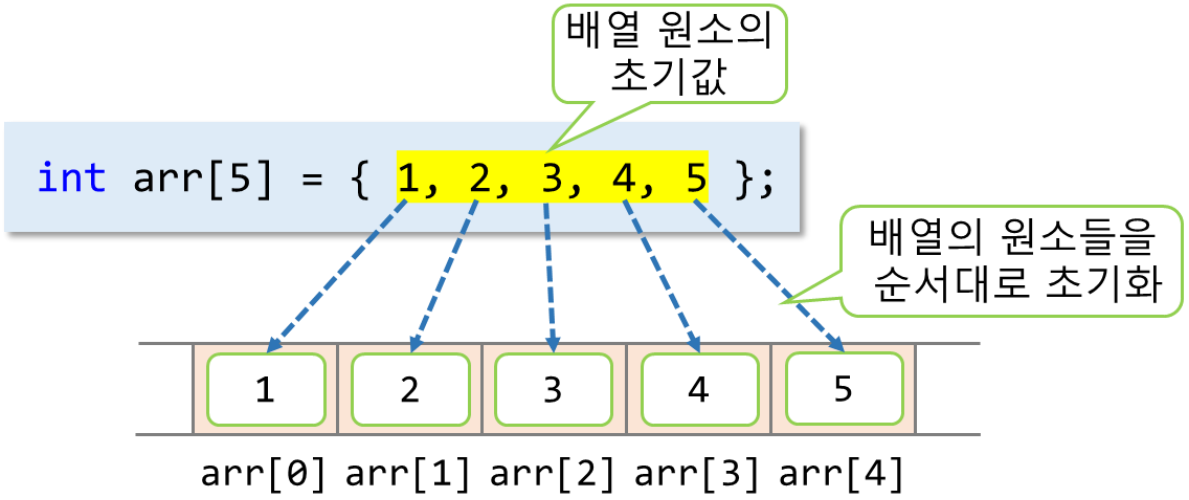
예제 7-2

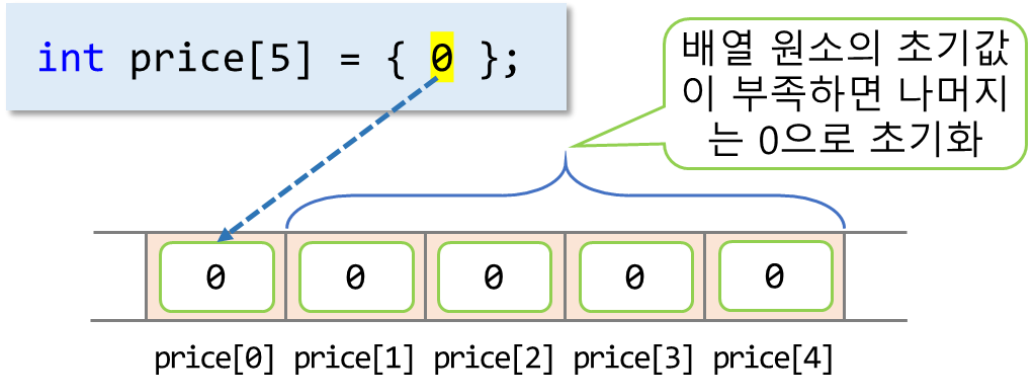


배열의 초기화 (1/2)

- { } 안에 배열 원소의 초기값을 콤마(,)로 나열한다.
- 배열의 0번째 원소부터 배열의 초기값이 나열된 순서대로 초기화한다.
- 초기값이 부족하면 나머지 원소는 0으로 초기화한다.

예제 7-3





배열의 초기화 (2/2)

예제 7-4

- 초기값을 원소의 개수보다 많으면 컴파일 에러가 발생한다.

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

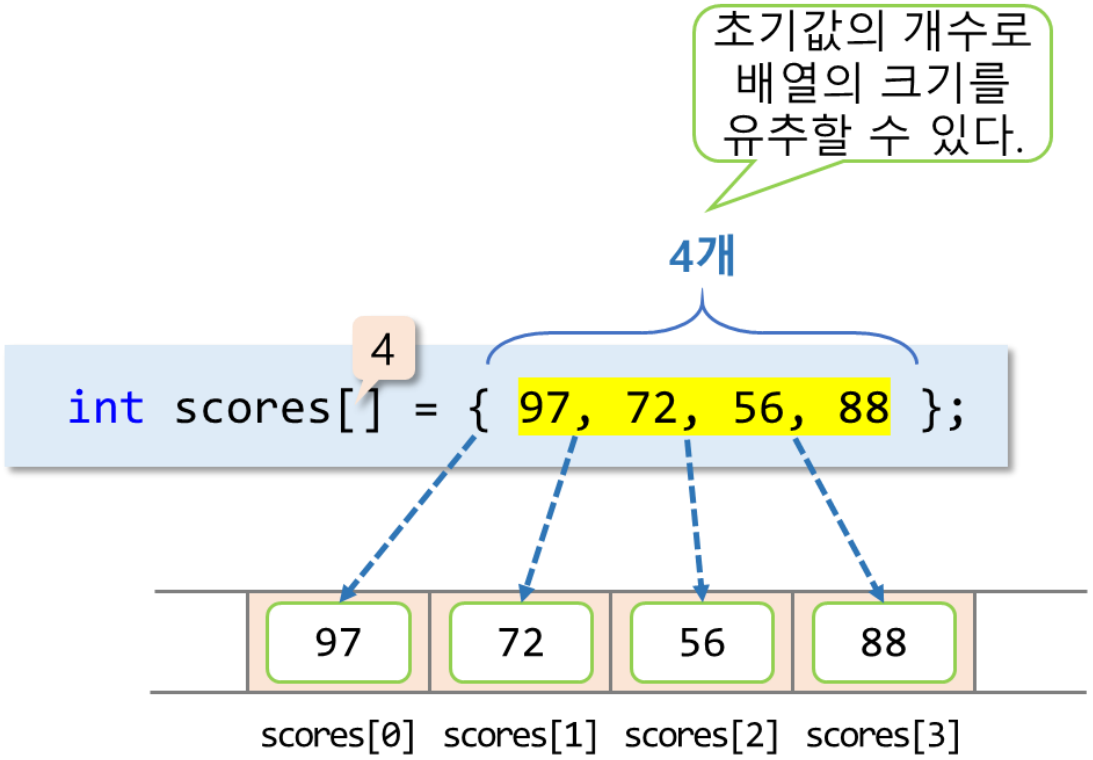
- { } 안을 비워 두면 컴파일 에러가 발생한다.

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

- 초기값을 지정하지 않고 배열의 크기를 생략하면 컴파일 에러가 발생한다.

```
int scores[];
```

- 배열의 초기값을 지정하는 경우에는 배열의 크기를 생략할 수 있다.



배열 원소의 사용

- 배열의 각 원소에 접근하려면 인덱스 또는 첨자를 이용한다.

예제 7-5

```
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.
```

- 배열은 주로 for문과 함께 사용된다.

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

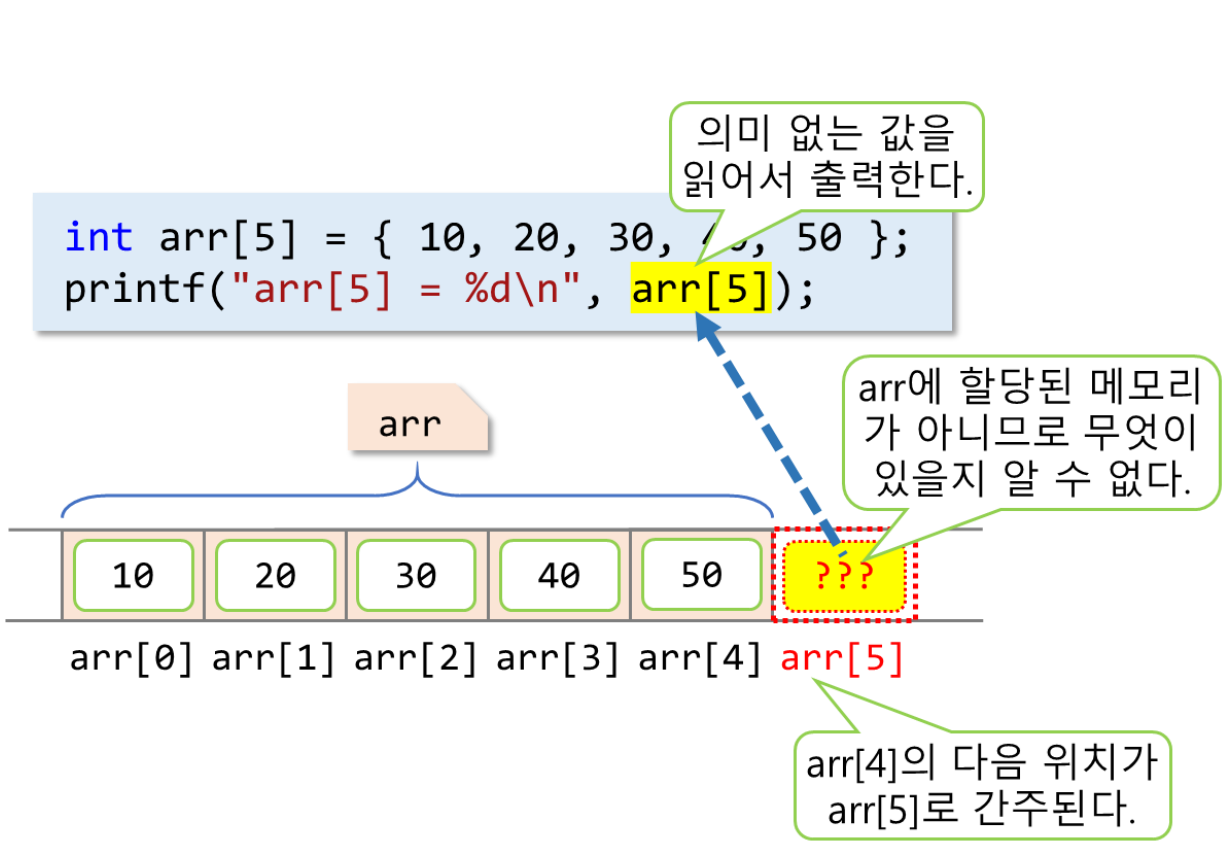
- 배열의 인덱스에는 변수나 변수를 포함한 수식을 사용할 수 있다.

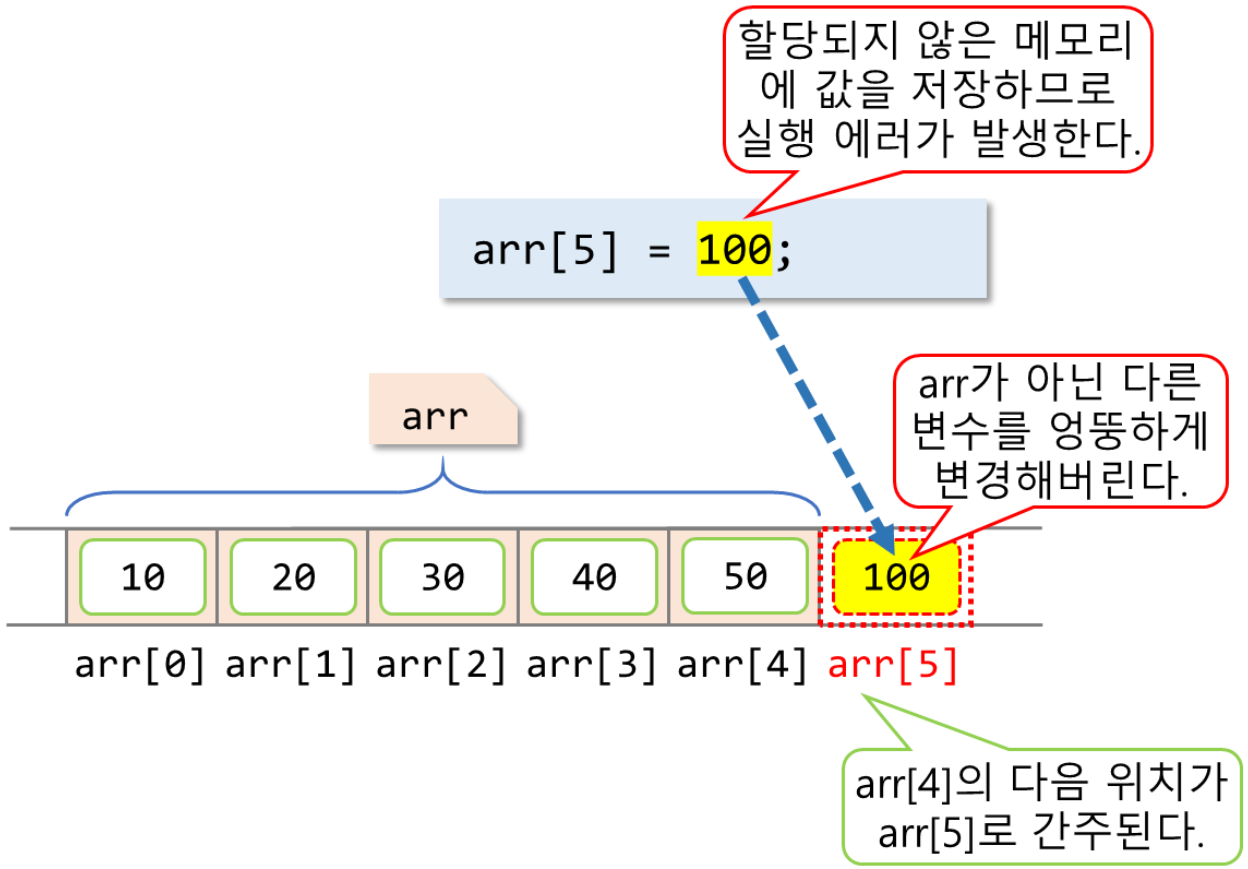
```
arr[i] = arr[i-1] * 2;         // 배열의 인덱스로 정수식을 사용할 수 있다.
```

배열 인덱스의 유효 범위

예제 7-6

- 배열의 인덱스는 항상 0~(배열의 크기 - 1)사이의 값이다.





배열의 복사

- 원소의 데이터형과 배열의 크기가 같은 경우에도 배열을 다른 배열에 대입할 수는 없다.

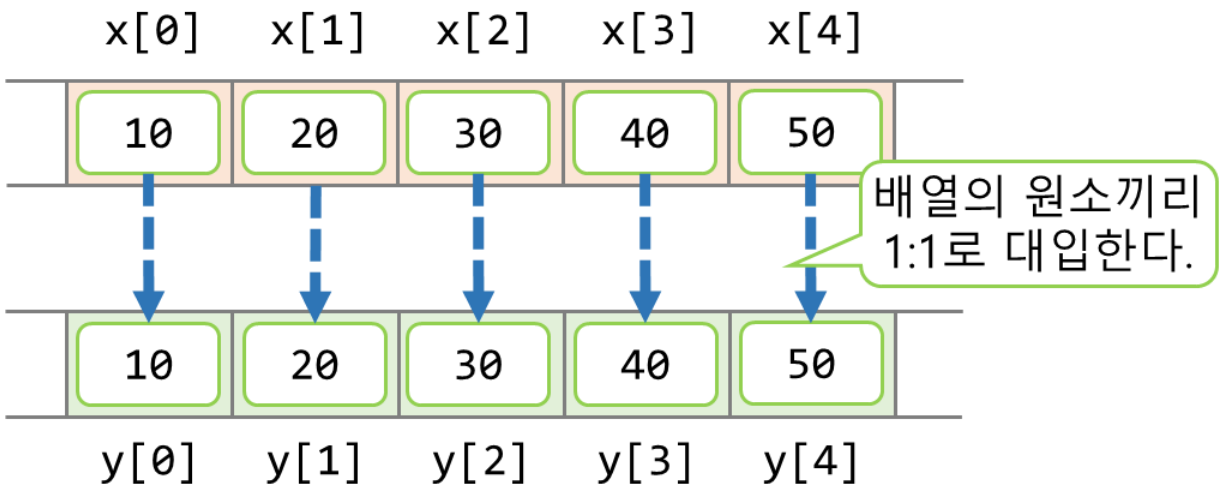
```
int x[5] = { 10, 20, 30, 40, 50 };
int y[5] = { 0 };
y = x;      // 컴파일 에러
```

배열에는
대입할 수 없다.

예제 7-7

- 배열을 복사하려면 원소끼리 1:1로 대입한다.

```
for (i = 0; i < 5; i++)
    y[i] = x[i];
```



배열의 비교

예제 7-8

- 두 배열이 같은지 비교하기 위해서 == 연산자로 직접 배열을 비교하면 안된다.

```
int x[5] = { 10, 20, 30, 40, 50 };
int y[5] = { 0 };
if (x == y)
    printf("두 배열이 같습니다\n");
```

배열의 주소를 비교한다.

- 인덱스 없이 배열 이름만 사용하면 배열의 시작 주소를 의미한다.

- 배열의 내용이 같은지 비교하려면 for문을 이용해서 원소끼리 비교해야 한다.

```
is_equal = 1;
for (i = 0; i < 5; i++)
{
    if (x[i] != y[i])
    {
        is_equal = 0;
        break;
    }
}
if (is_equal == 1)
    printf("두 배열이 같습니다.\n");
```

배열이 같은지를 나타내는 변수

배열의 원소끼리 비교한다.

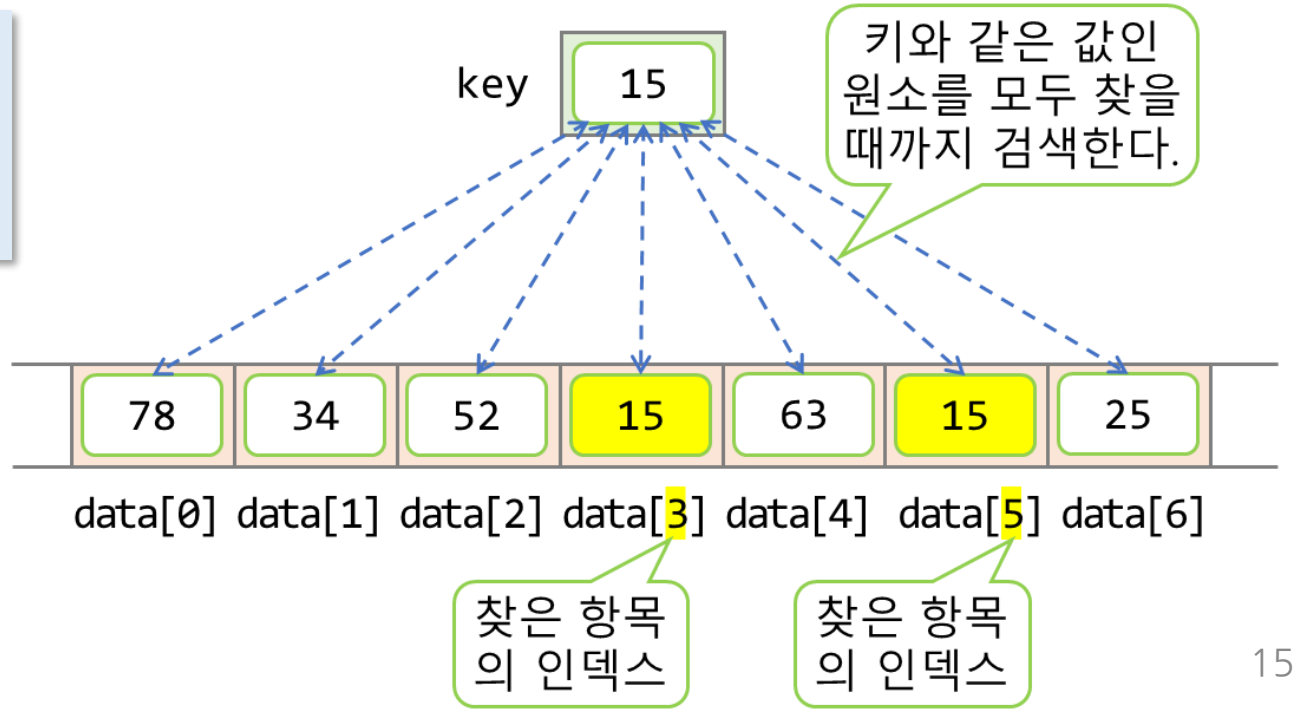
서로 다른 원소가 있으면 더 이상 비교할 필요가 없다.

모든 원소가 같으면 is_equal은 1

배열의 탐색 (1/2)

- 주어진 데이터 중에서 특정 값을 가진 항목을 찾는 기능 예제 7-9
- **순차 탐색(sequential search)**
  - 배열의 0번째 원소부터 순서대로 탐색키와 비교
- 탐색키와 일치하는 항목을 모두 찾아야 하는 경우

```
for (i = 0; i < size; i++)
if (data[i] == key)
  printf("찾은 원소의 인덱스: %d\n", i);
```



배열의 탐색 (2/2)

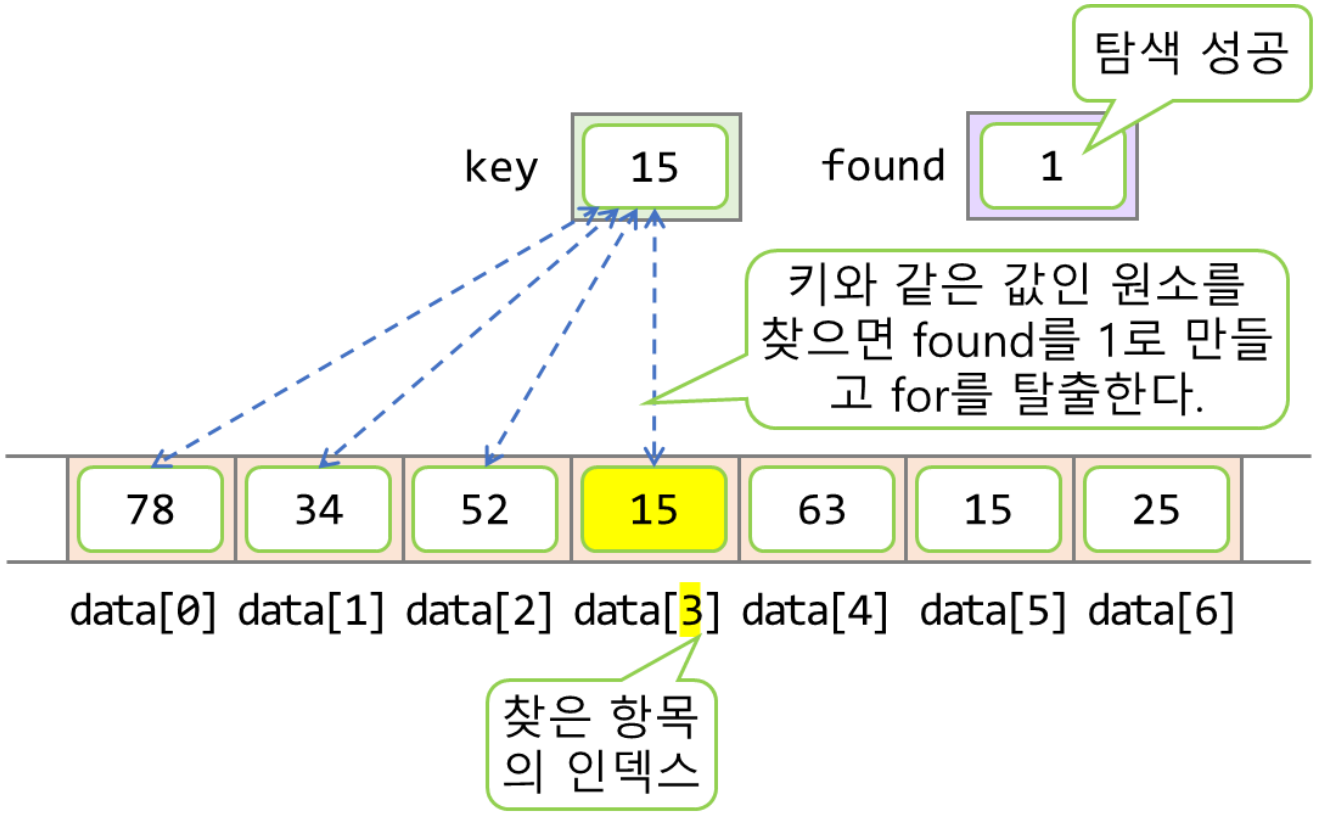
- 탐색키와 일치하는 첫 번째 항목만 찾는 경우
  - 탐색키와 같은 값을 가진 원소를 찾으면 break로 for를 탈출한다.

예제 7-10

```
found = 0;
for (i = 0; i < size; i++) {
  if (data[i] == key) {
    found = 1;
    break;
  }
}
if (found == 1)
  printf("찾은 원소의 인덱스: %d\n", i);
else
  printf("탐색 실패\n");
```

탐색이 성공하면 1, 실패하면 0

탐색 성공 시 for 탈출



2진 탐색

- 배열을 정렬한 상태에서 탐색을 수행하는 방법
- 표준 C 라이브러리의 2진 탐색 함수

```
void* bsearch(const void *key, const void *ptr, size_t count, size_t size,
    int(*comp)(const void*, const void*));        // <stdlib.h>을 포함해야 한다.
```

배열의 정렬

- 주어진 데이터를 조건에 따라서 나열하는 기능
  - 오름차순(ascending order) 정렬 : 크기가 커지는 순서로 나열
  - 내림차순(descending order) 정렬 : 크기가 작아지는 순서로 나열
- 데이터가 정렬되어 있으면 데이터에 대한 여러 가지 작업이 간단해진다.
  - 최소값을 찾거나, 최대값을 찾는 작업은 배열의 0번 원소나 마지막 원소를 가져오면 된다.
  - 2진 탐색을 할 수 있으므로 빠른 탐색이 가능하다.
- **선택 정렬(selection sort)**
  - 전체 배열의 원소 중 가장 작은 값을 찾아서 배열의 첫 번째 위치로 옮기고, 그 다음 작은 값을 찾아서 배열의 두 번째 위치로 옮기는 식으로 정렬한다.

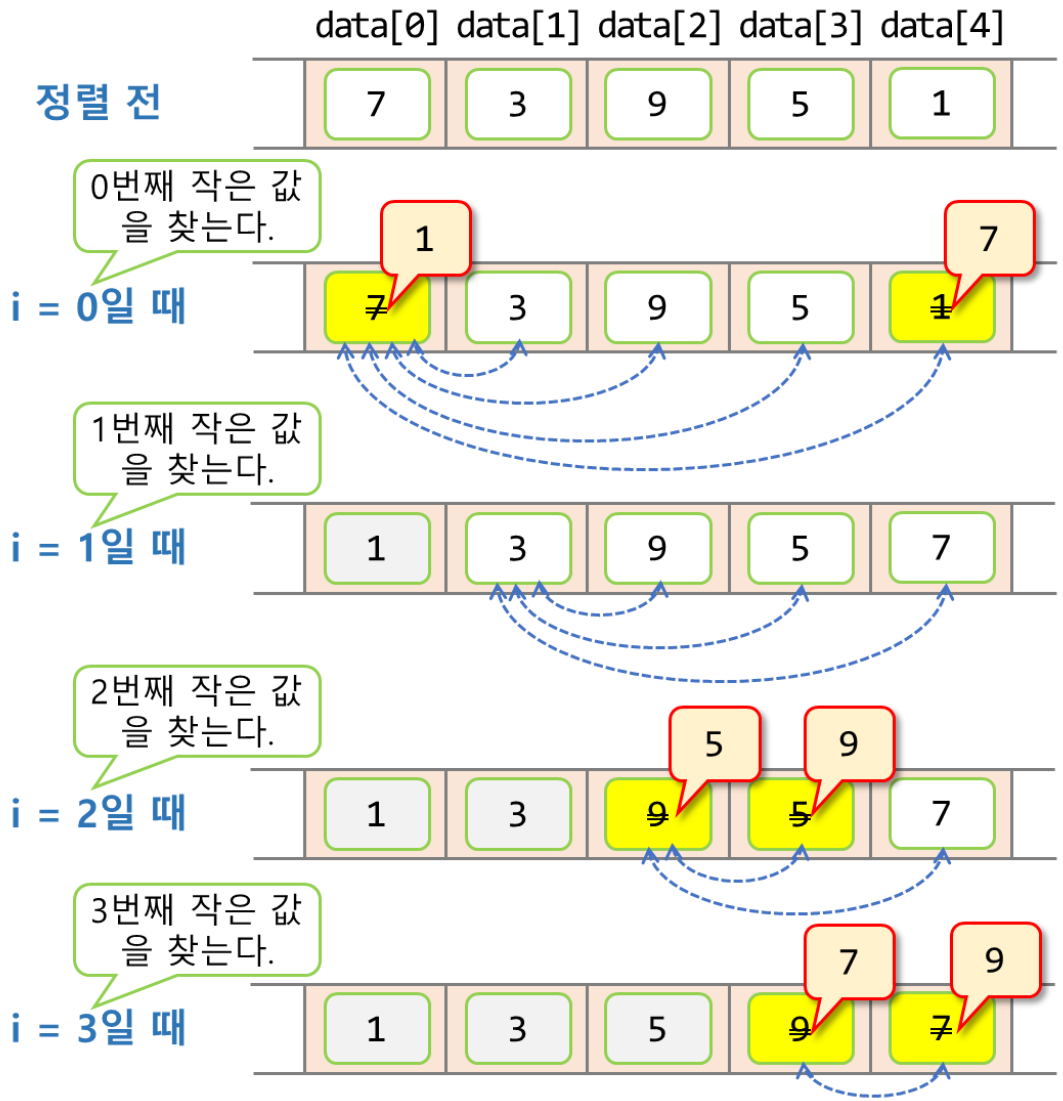
선택 정렬 (1/2)

예제 7-11

```
for (i = 0; i < SIZE - 1; i++)   // 0~(i-1)까지는 정렬된 상태이다.
{
    index = i;
    // data[i]~data[SIZE-1]중에서 가장 작은 원소의 인덱스를 index에 저장한다.
    for (j = i + 1; j < SIZE; j++)
    {
        if (data[index] > data[j])
            index = j;
    }

    // i번째 원소를 index에 있는 원소와 맞바꾼다.
    if (i != index)
    {
        temp = data[i];
        data[i] = data[index];
        data[index] = temp;
    } // i번째 원소가 i번째로 작은 값이 된다.
}
```

선택 정렬 (2/2)



data[0]과 data[1]~data[4]를 비교해서 data[0]을 가장 작은 data[4]와 바꾼다.

data[1]과 data[2]~data[4]를 비교해서 data[1]이 가장 작으므로 그대로 둔다.

data[2]과 data[3]~data[4]를 비교해서 data[2]를 가장 작은 data[3]과 바꾼다.

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

퀵 정렬

- 분할 정복 알고리즘으로 정렬 수행
- 표준 C 라이브러리의 퀵 정렬 함수

```
void qsort(void *ptr, size_t count, size_t size,
    int(*compare)(const void *, const void *));          // <stdlib.h>을 포함해야 한다.
```

다차원 배열의 개념 (1/3)

- 2차원 배열
  - 행(row)과 열(column)의 개념으로 이해할 수 있다.

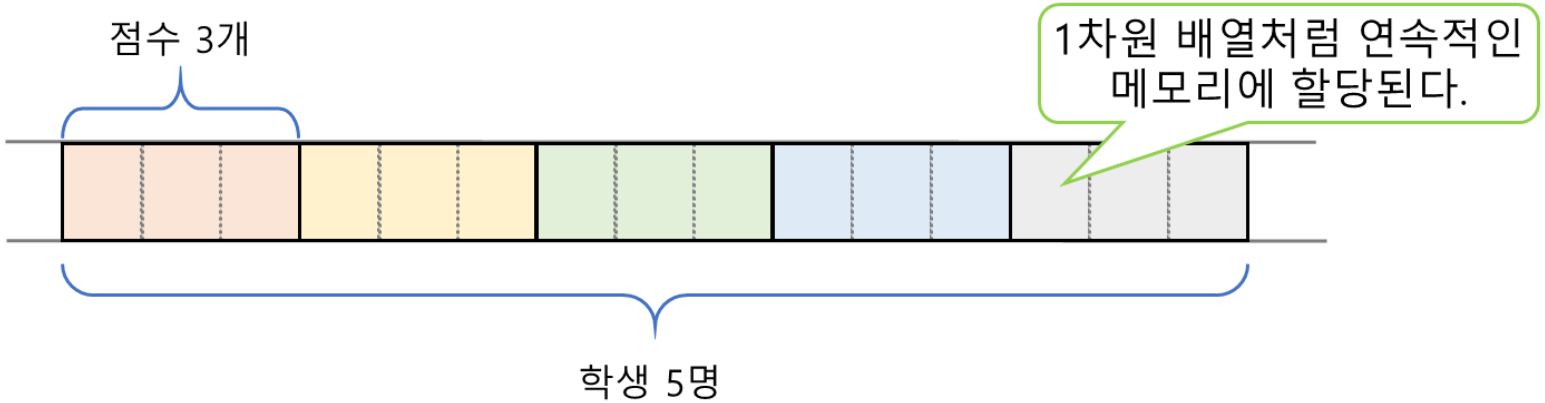
```
int scores[5][3];
```

행 열

번호	중간고사 점수	기말고사 점수	과제 점수
0	scores[0][0]	scores[0][1]	scores[0][2]
1	scores[1][0]	scores[1][1]	scores[1][2]
2	scores[2][0]	scores[2][1]	scores[2][2]
3	scores[3][0]	scores[3][1]	scores[3][2]
4	scores[4][0]	scores[4][1]	scores[4][2]

int 3개인 배열

int 3개인 배열이 모두 5개 필요하다.

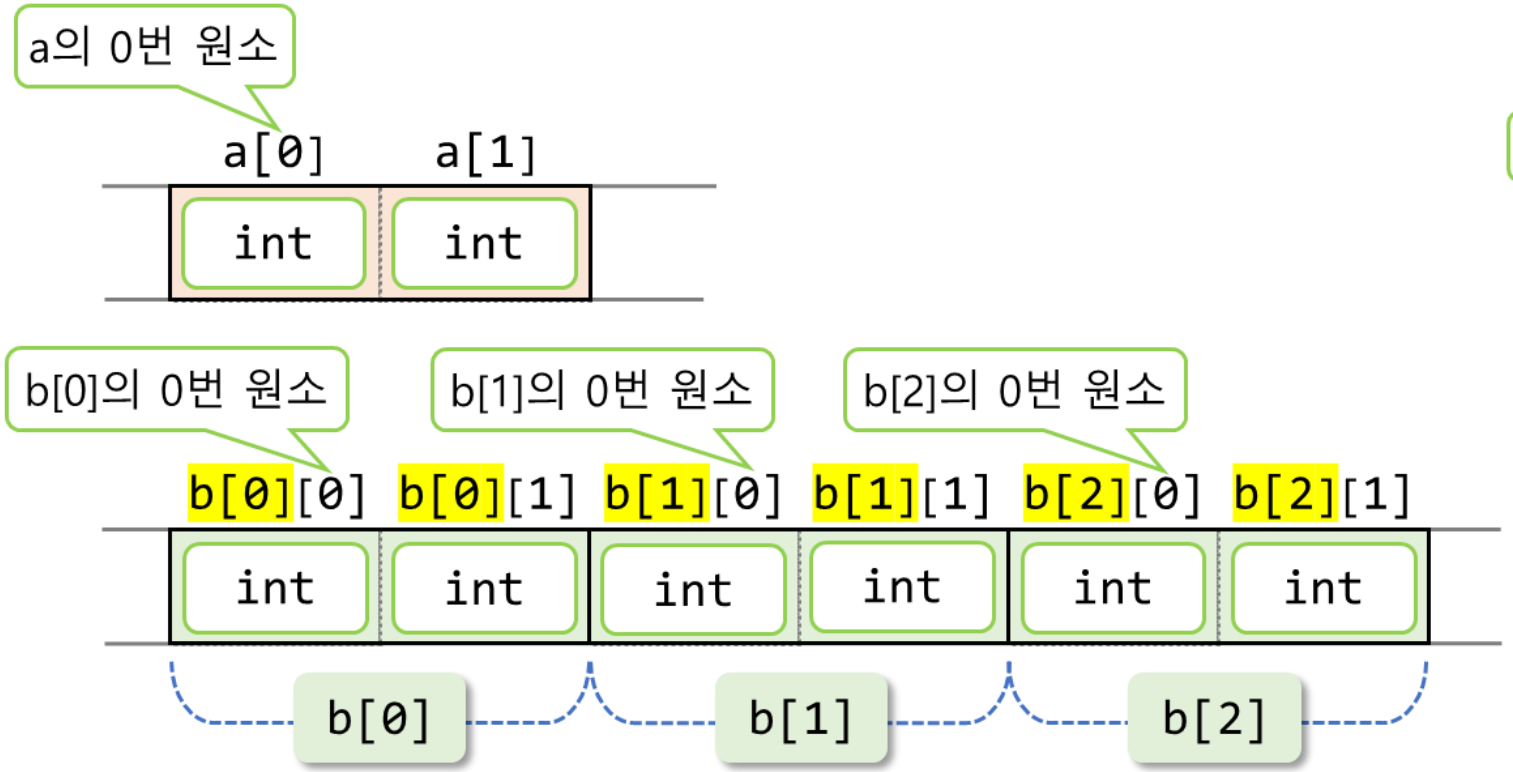


다차원 배열의 개념 (2/3)

```
int a[2];
int b[3][2];
```

int가 2개인 배열

int[2]가 3개인 배열



배열명 인덱스

```
a[i]
```

배열명 인덱스

```
b[i][j]
```

다차원 배열의 개념 (3/3)

- 배열 이름 바로 다음의 [ ] 안에 나오는 것이 배열의 크기이고, 나머지 부분은 배열의 원소형으로 보면 된다.

```
int a[2];          // 1차원 배열 => 크기는 2이고, 원소형은 int
int b[3][2];       // 2차원 배열 => 크기는 3이고, 원소형은 int[2]
int c[4][3][2];    // 3차원 배열 => 크기는 4이고, 원소형은 int[3][2]
int d[5][4][3][2]; // 4차원 배열 => 크기는 5이고, 원소형은 int[4][3][2]
```

- 전체 원소의 개수를 기준으로 판단한다.

```
int x[100];              // 원소가 100개인 배열
int y[200][100];         // 원소가 100×200개인 배열
int z[300][200][100];    // 원소가 100×200×300개인 배열
```

2차원 배열의 선언

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

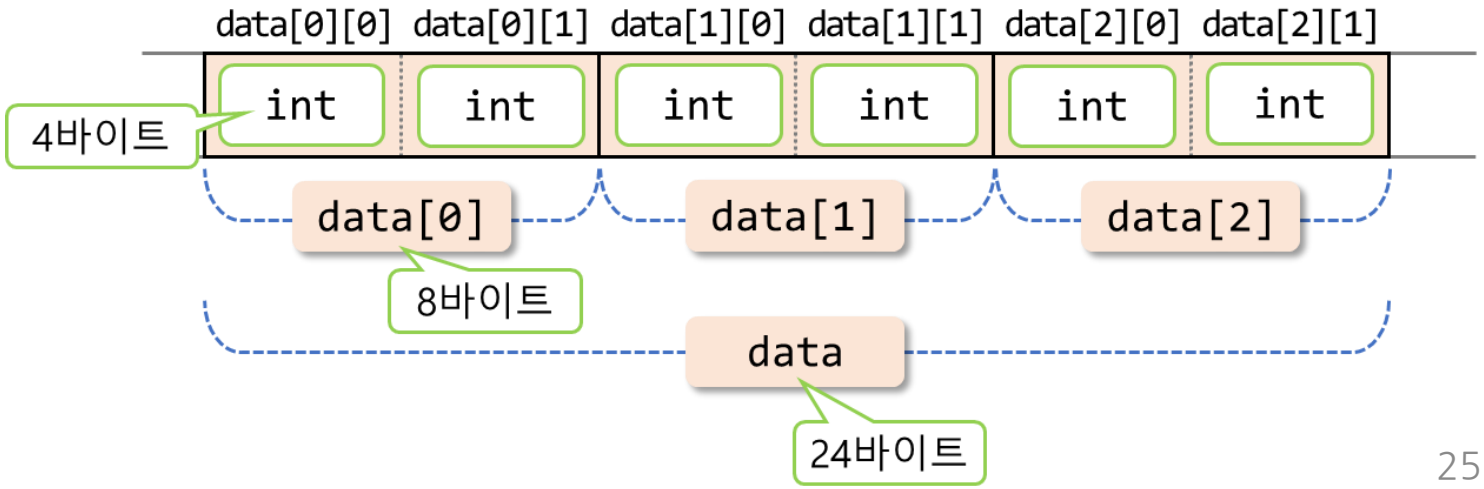
형식 데이터형 배열명[행크기][열크기];

사용예

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

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

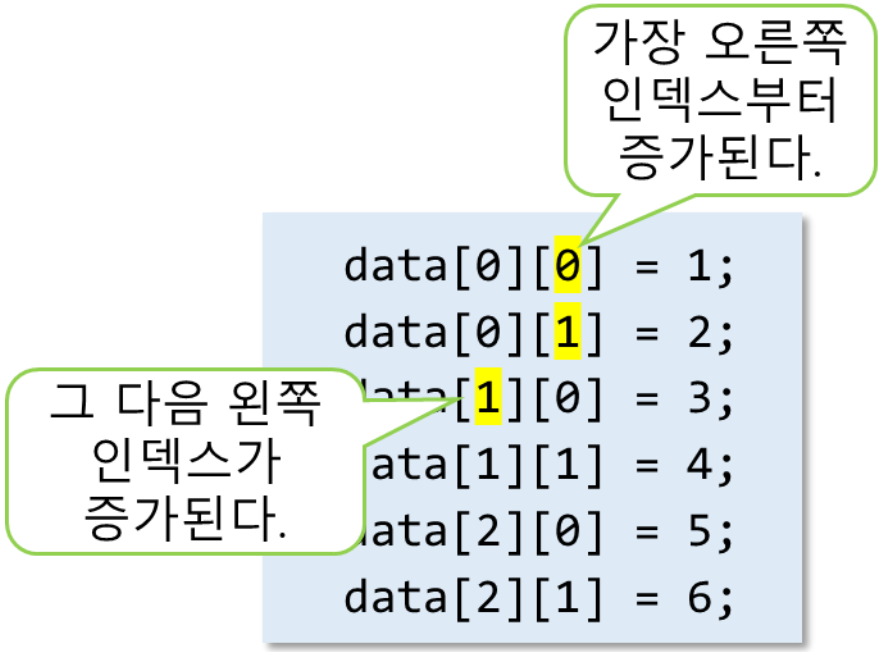
배열의 크기를 매크로 상수로 지정한다.

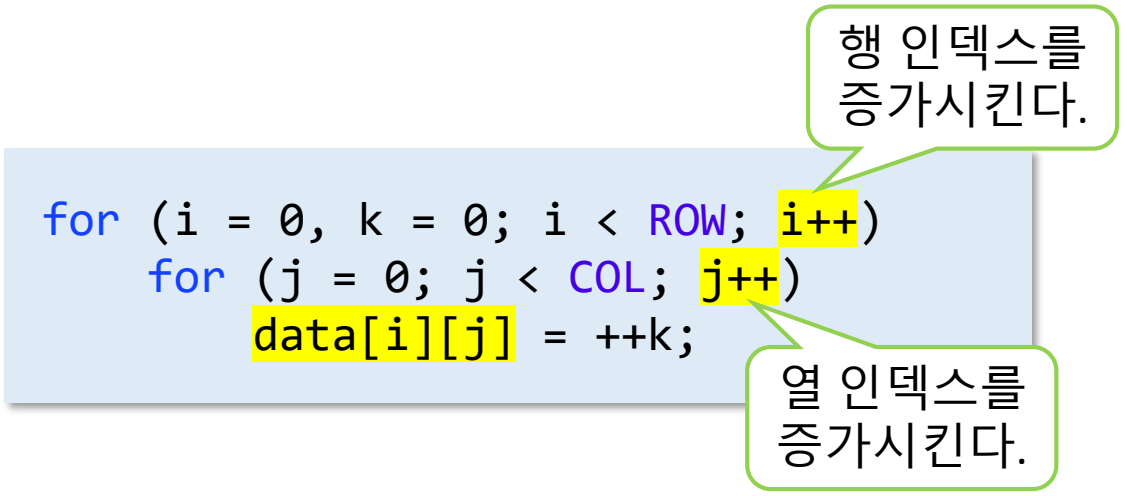


2차원 배열의 사용

예제 7-12

- 2차원 배열은 원소에 접근할 때 인덱스를 2개 사용한다.
  - 인덱스를 여러 개 사용할 때는 가장 오른쪽 인덱스부터 증가되고, 가장 오른쪽 인덱스가 모두 증가되면 다시 그 왼쪽에 있는 인덱스가 증가된다.





2차원 배열의 초기화

예제 7-13

- 초기값을 열 크기의 개수만큼씩 { }로 묶어서 다시 { } 안에 나열한다.

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

- 1차원 배열처럼 { } 안에 값만 나열할 수도 있다.

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

- 초기값을 생략하면 나머지 원소를 0으로 초기화한다.

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

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

- 초기값을 지정하는 경우에 배열의 행 크기를 생략할 수 있다.

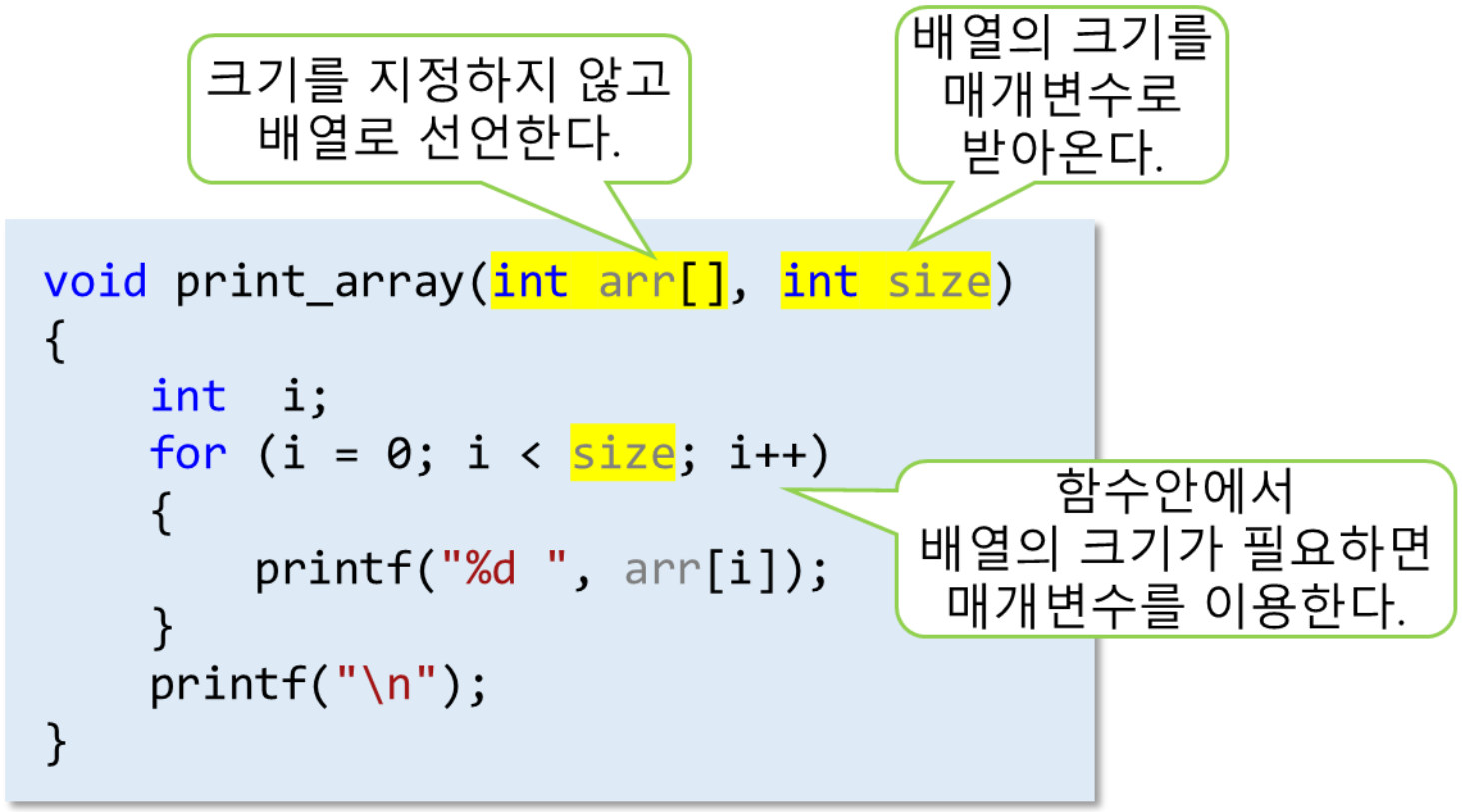
3

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

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

함수의 인자로 배열 전달하기

- 함수의 매개변수로 배열을 선언할 때는 배열의 크기를 생략한다.
- 배열의 크기를 함수의 매개변수로 받아와야 한다.




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

예제 7.-14,15

학습과제

- 학습과제
  - 7장 함수 교재 내용 이해하고 연습문제 풀이하여 그 결과를 메일로 보내기
  - Exercise 7
    - 페이지 365~369페이지

  - 메일 주소 : wykim@gw.ac.kr
  - 메일 제목 : 학번_이름_11차~12차 과제