7장 배열

박 종 혁 교수
서울과학기술대학교 컴퓨터공학과
UCS Lab
Tel: 970-6702
Email: jhpark1@seoultechackr

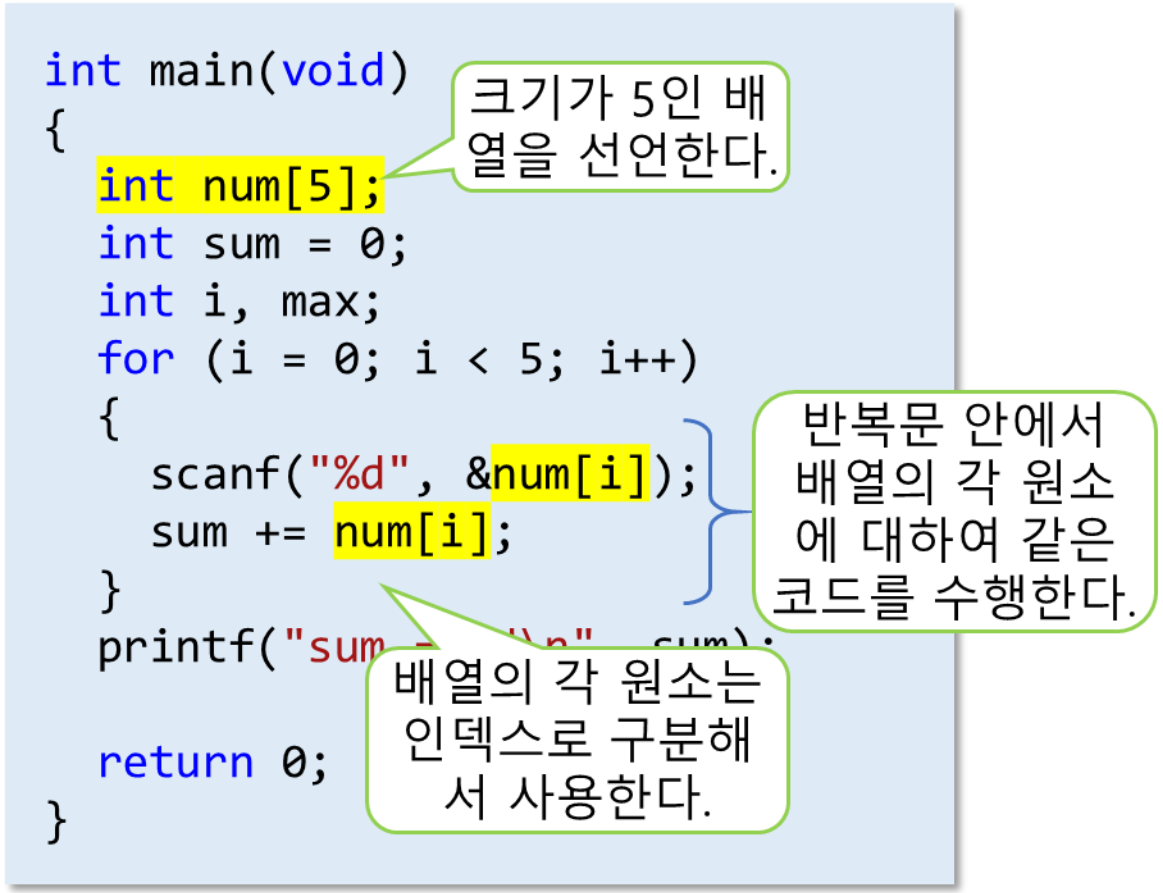
목차

❖ 배열의 기본

- 배열의 개념
- 배열의 선언
- 배열의 초기화
- 배열의 사용

배열의 개념

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

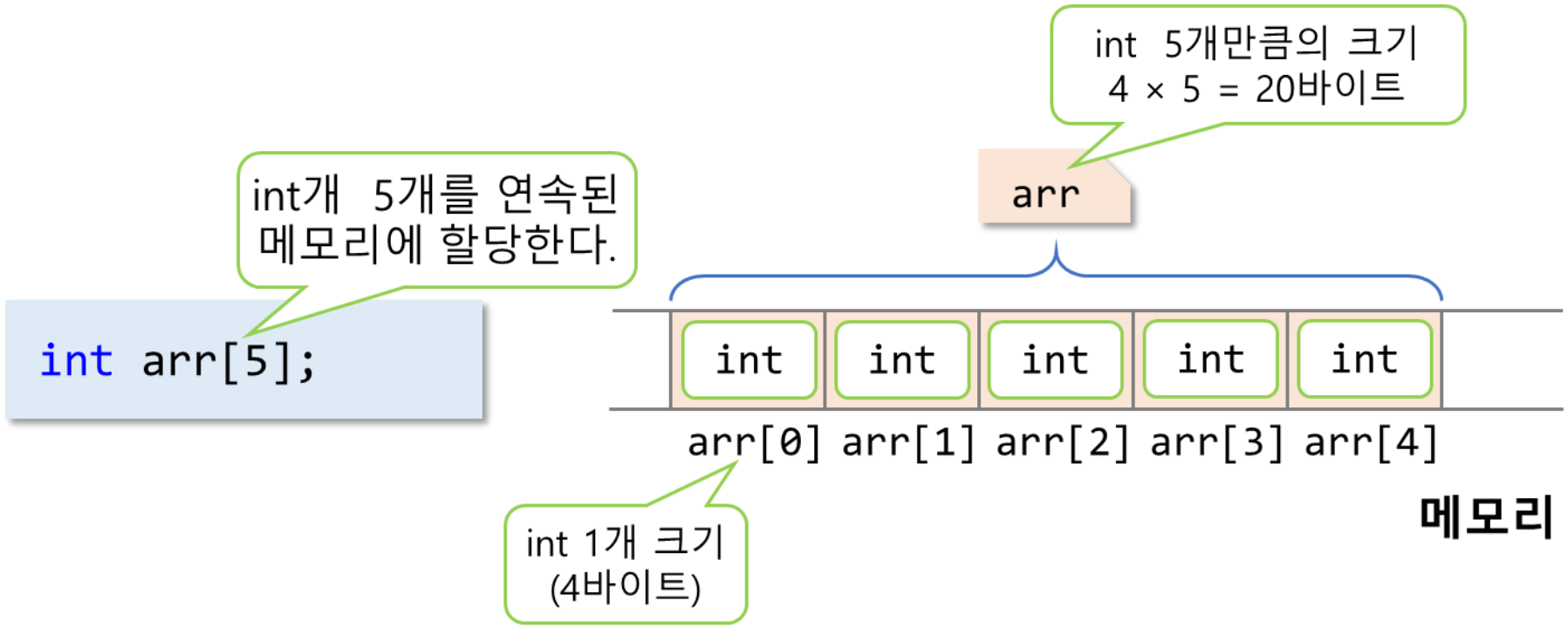


배열의 선언

형식 데이터형 배열명[크기];

사용예

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



첨자

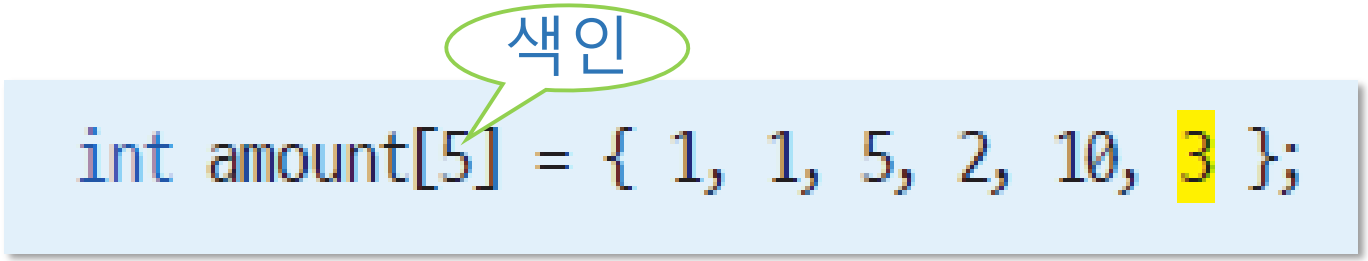
❖ 사용 예

```
int     i, a[N];
```

- 여기서 N은 기호 상수이고,
- 하나의 배열 원소를 참조하기 위해서는 첨자 i에 적당한 값을 할당한 후 a[i] 수식을 사용하면 됨
- 이때, i는 0보다 크거나 같고 N -1보다 작거나 같아야함
- i가 이 범위를 벗어나는 값을 갖는다면, a[i]를 접근할 때 실행 시간 오류가 발생함

첨자

❖ A가 배열이면, a의 원소를 접근할 때 a[expr]와 같이 씀



❖ 여기서, expr은 정수적 수식이고,
Expr을 a의 첨자, 또는 색인이라고 함

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
```

예제 : 배열의 바이트 크기와 크기 구하기

```c
int main(void)
{
    int arr[5];                 // 크기가 5인 배열 선언
    int byte_size = 0;          // 배열의 바이트 크기를 저장할 변수
    int size = 0;               // 배열의 크기를 저장할 변수
    int i;

    byte_size = sizeof(arr);  // 배열의 바이트 크기
    printf("배열의 바이트 크기: %d\n", byte_size);

    size = sizeof(arr) / sizeof(arr[0]);          // 배열의 크기(원소의 개수)
    printf("배열의 크기: %d\n", size);

    for (i = 0; i < size; i++)
        arr[i] = 0;

    return 0;
}
```

실행결과

```
배열의 바이트 크기: 20
배열의 크기: 5
```

배열의 크기를 구해서 사용하는 이유

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

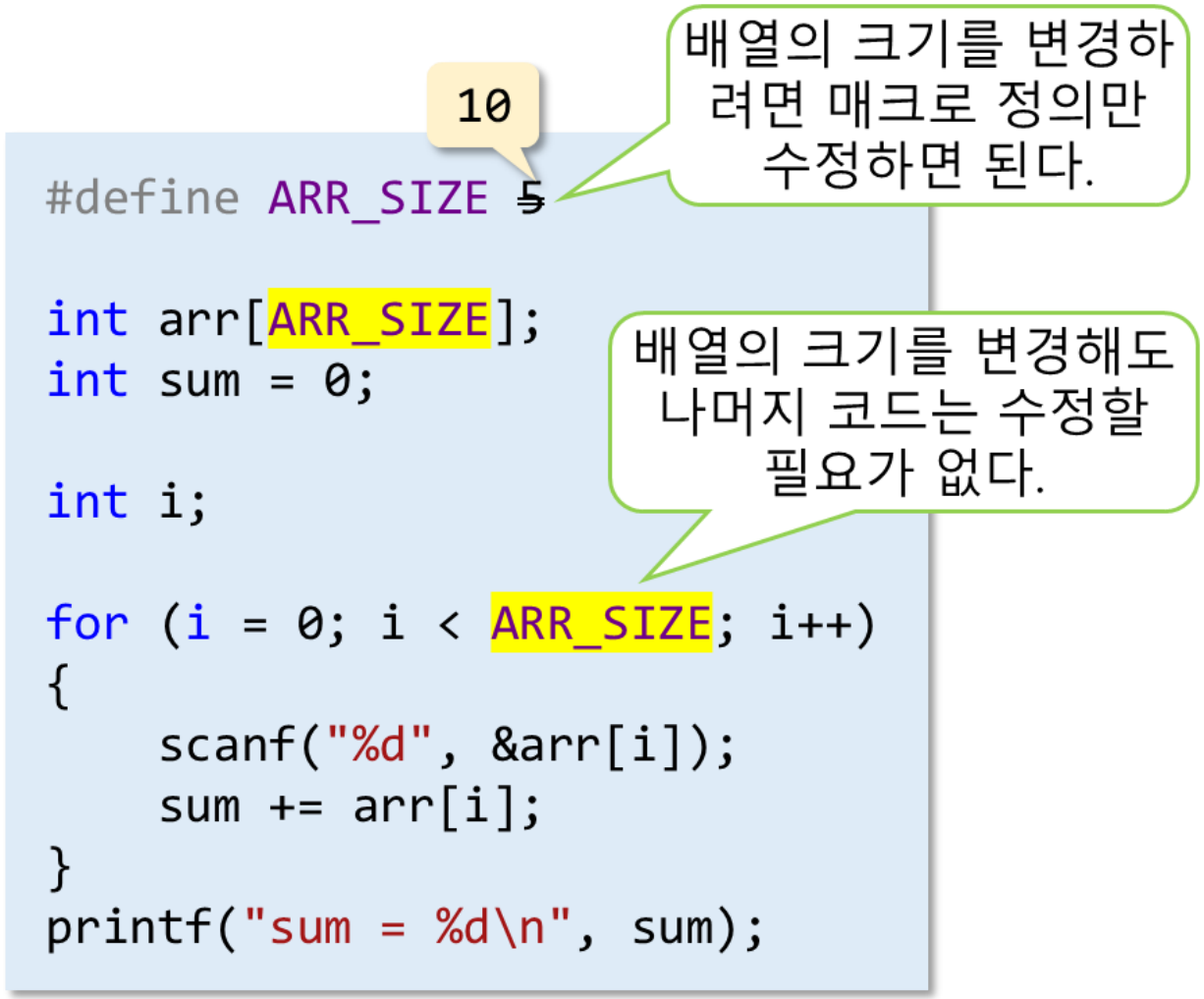
```
int arr[5];
int sum = 0;
int size =
  sizeof(arr)/sizeof(arr[0]);
int i;

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 : 매크로 상수로 배열의 크기를 지정하는 경우

```c
#define ARR_SIZE 5 // 배열의 크기로 사용할 매크로 상수의 정의

int main(void)
{
    int arr[ARR_SIZE];       // 배열의 크기를 매크로 상수로 지정할 수 있다.
    int i;

    for (i = 0; i < ARR_SIZE; i++)     // 배열의 크기가 필요하면 ARR_SIZE 이용
        arr[i] = 0;

    printf("arr = ");
    for (i = 0; i < ARR_SIZE; i++)     // 배열의 크기가 필요하면 ARR_SIZE 이용
        printf("%d ", arr[i]);
    printf("\n");

    return 0;
}
```

실행결과

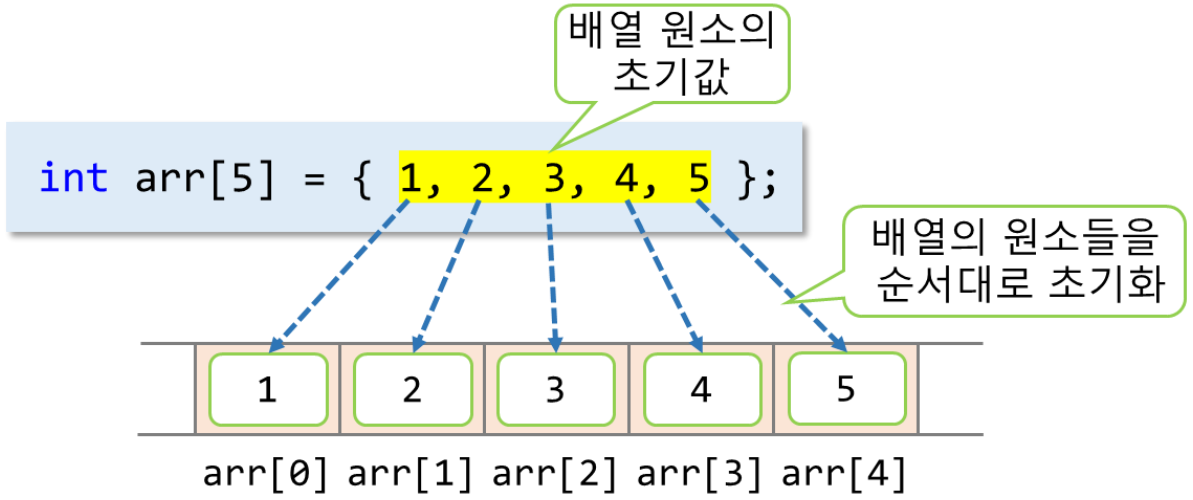
```
arr = 0 0 0 0 0
```

배열의 초기화 (1/4)

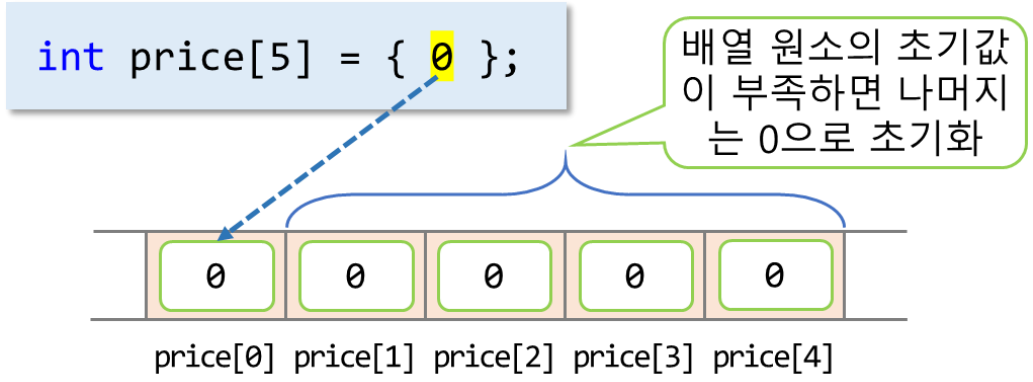
- 배열은 자동, 외부, 정적 기억영역 클래스는 될 수 있지만, 레지스터는 될 수 없음
- 전통적인 C에서는 외부와 정적 배열만 배열 초기 자를 사용하여 초기화할 수 있음
- ANSI C에서는 자동 배열도 초기화될 수 있음

배열의 초기화 (2/4)

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



- 초기값이 부족하면 나머지 원소는 0으로 초기화한다



예제 : 가장 기본적인 배열의 초기화

```
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
```

예제 : 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
```

배열의 초기화 (3/4)

- 초기값을 원소의 개수보다 많으면 컴파일 에러가 발생한다

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

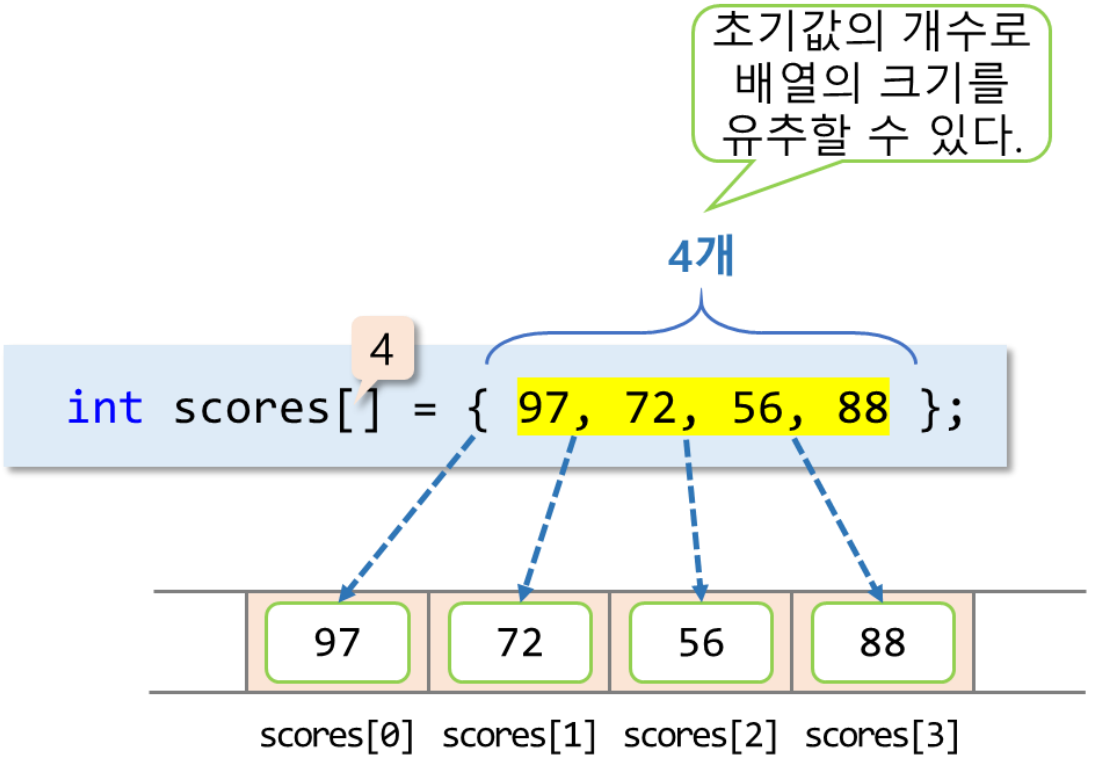
- { } 안을 비워 두면 컴파일 에러가 발생한다

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

- 초기값을 지정하지 않고 배열의 크기를 생략하면 컴파일 에러가 발생한다

```
int scores[];
```

- 배열의 초기값을 지정하는 경우에는 배열의 크기를 생략할 수 있다



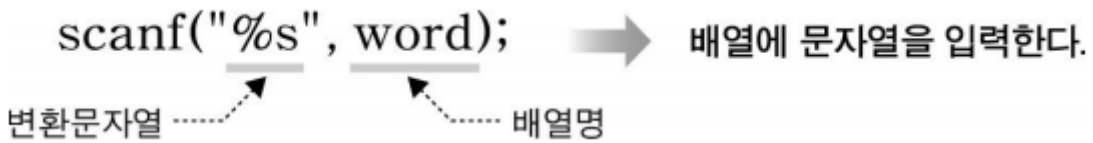
배열의 초기화 (4/4)

- 초기자 목록이 초기화되는 배열 원소 개수보다 적다면, 나머지 원소들은 0으로 초기화됨

```
int a[100] = {10} ;
→ a[0] = 10, a[1] = 0, a[2] = 0, , a[99] = 0
```

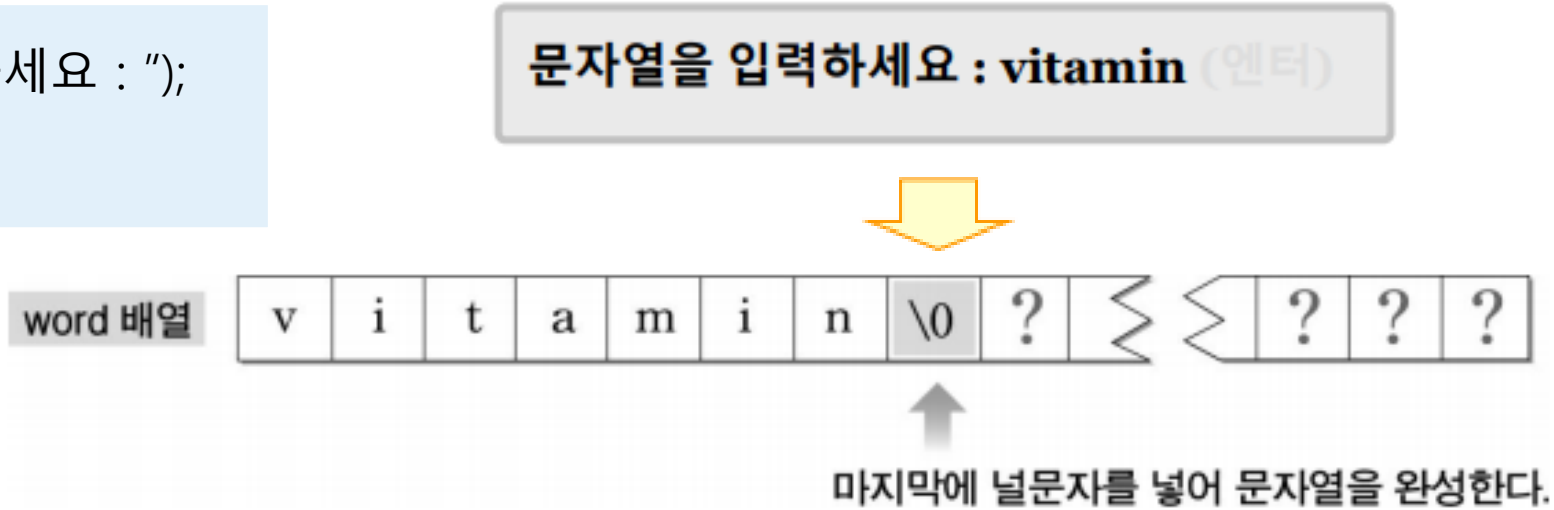
scanf 함수를 사용한 문자열의 입력 (1/2)

❖문자배열에 문자열을 입력 받을 때는 %s변환문자열과 배열명을
- scanf함수의 전달인자로 준다



❖scanf함수로 문자열을 입력 받으면 널문자를 자동으로 채워준다
- word배열에 vitamin을 입력 받은 경우

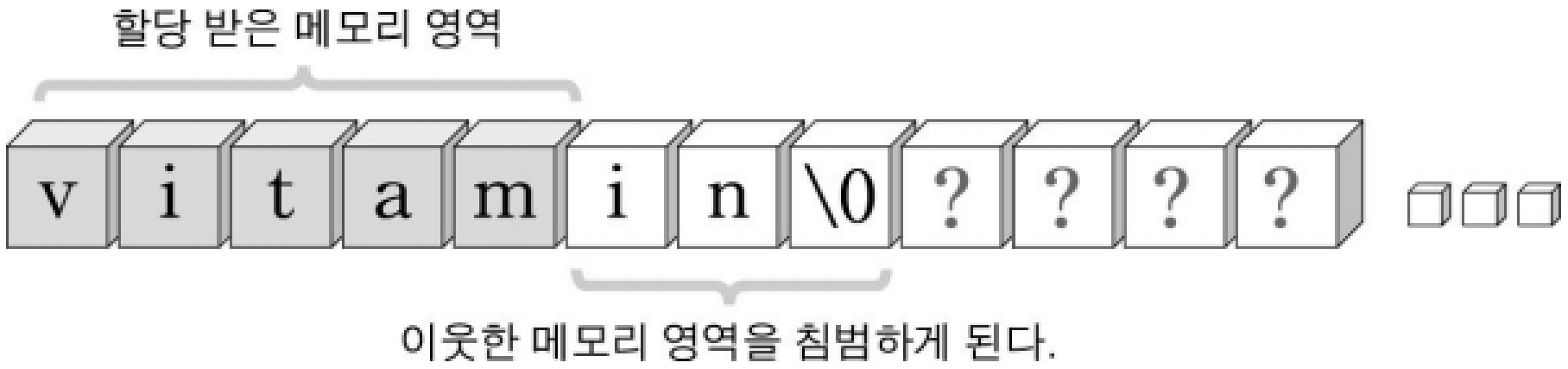
```
printf("문자열을 입력하세요 : ");
scanf("%s", word);
```



scanf 함수를 사용한 문자열의 입력 (2/2)

- 배열의 크기보다 입력되는 문자열의 크기가 더 크면 할당되지 않은 기억공간을 침범하므로 주의해야 한다

char word[5]; //이곳에 "vitamin"을 입력 받는다면...



배열 원소의 사용

- 배열의 각 원소에 접근하려면 인덱스 또는 첨자를 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.
```

- 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.
```

예제 : 배열의 원소가 변수로서 사용되는 경우

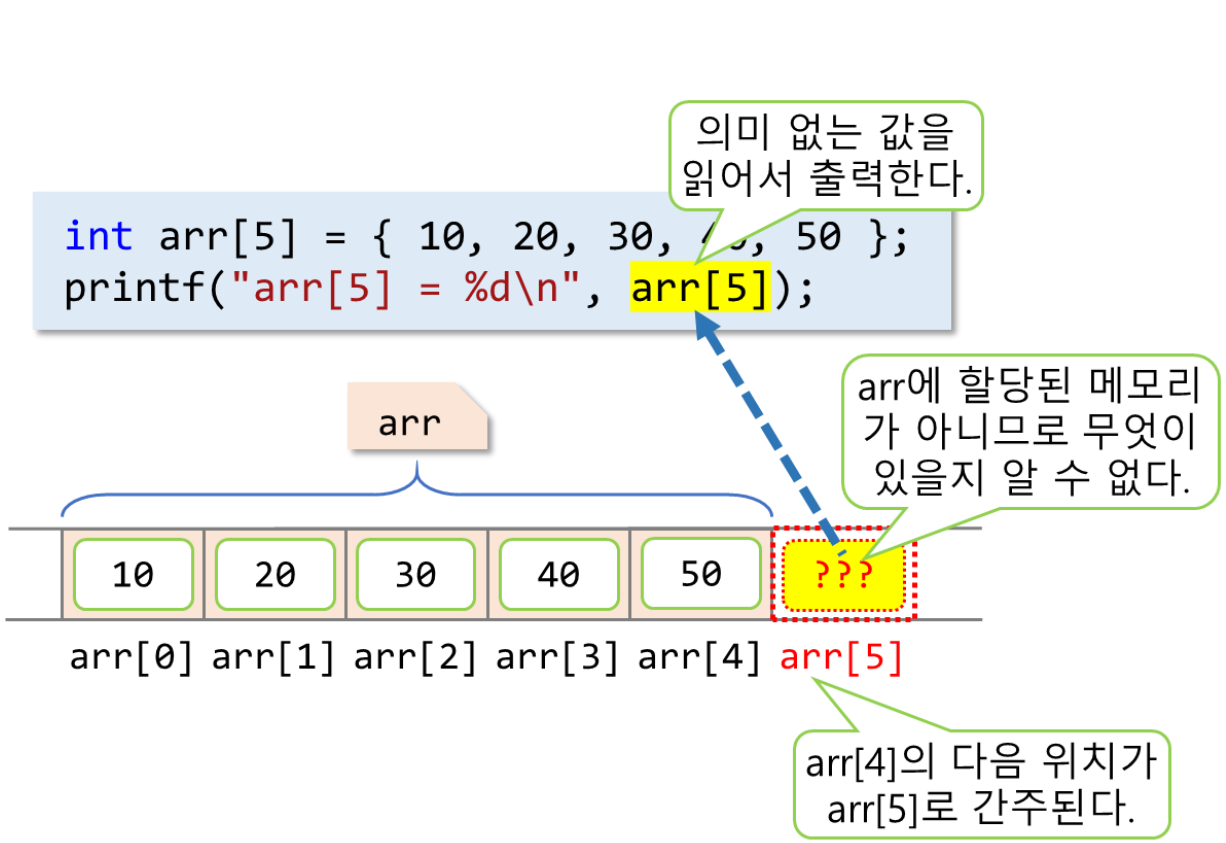
```c
#define ARR_SIZE 5

int add(int a, int b) { return a + b; }

int main(void)
{
  int arr[ARR_SIZE] = { 0 }; // 배열 전체를 0으로 초기화
  int i;

  arr[0] = 5;
  arr[1] = arr[0] + 10;       // 배열의 원소를 수식에 이용
  arr[2] = add(arr[0], arr[1]);      // 함수의 인자로 전달
  printf("정수를 2개 입력하세요: ");
  scanf("%d %d", &arr[3], &arr[4]); // 배열의 원소로 입력

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

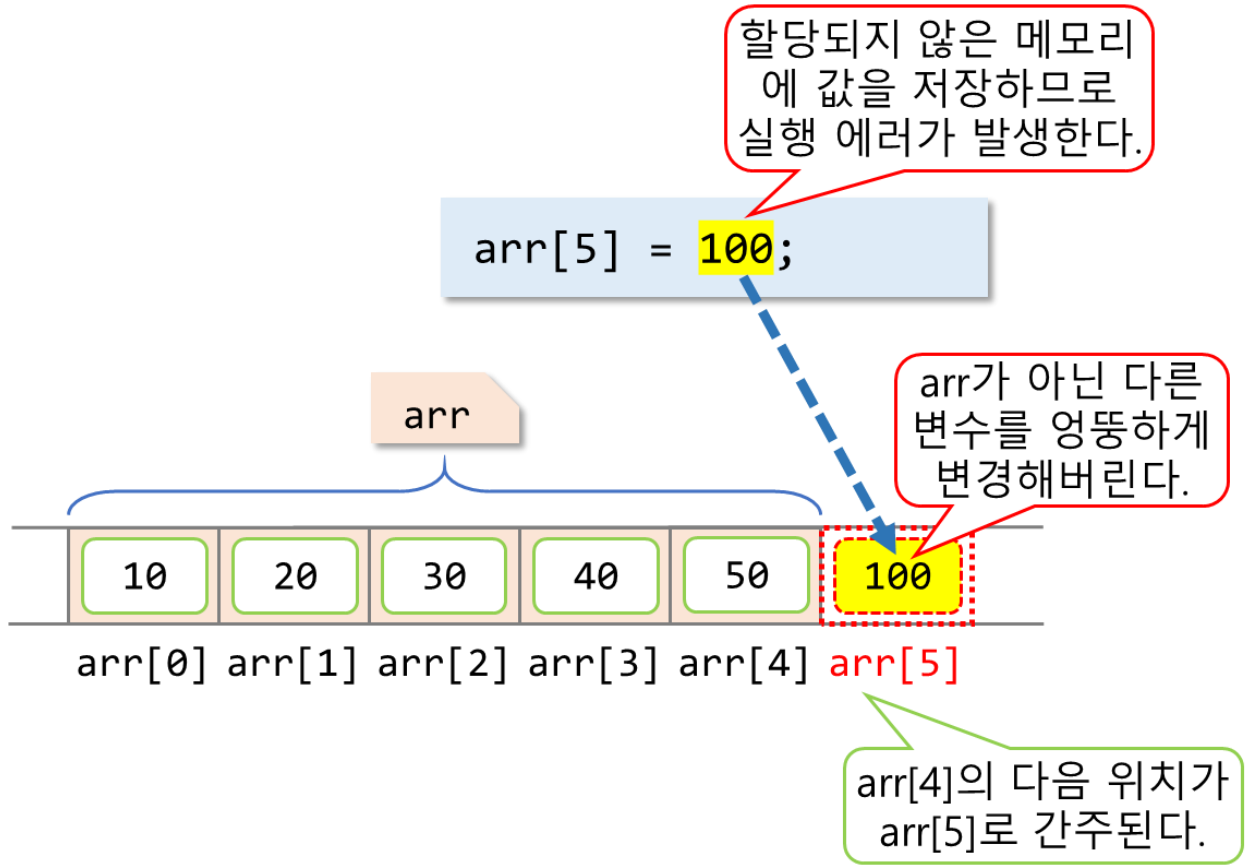
실행결과

```
정수를 2개 입력하세요: 55 66
5 15 20 55 66
```

배열 인덱스의 유효 범위

- 배열의 인덱스는 항상 0~(배열의 크기 - 1)사이의 값이다





예제 : 잘못된 인덱스를 사용하는 경우

```
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

배열의 복사

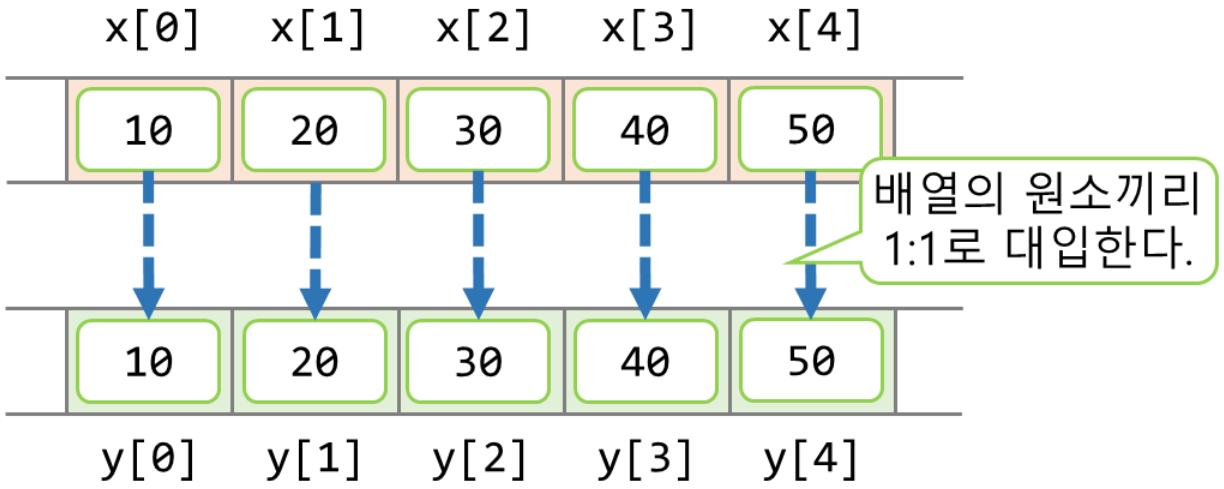
- 원소의 데이터형과 배열의 크기가 같은 경우에도 배열을 다른 배열에 대입할 수는 없다

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

배열에는
대입할 수 없다

- 배열을 복사하려면 원소끼리 1:1로 대입한다

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



예제 : 배열의 복사

```
03  int main(void)
04  {
05      int x[5] = { 10, 20, 30, 40, 50 };
06      int y[5] = { 0 };
07      int i;
08
09      //y = x;          // 배열을 다른 배열에 대입하면 컴파일 에러
10
11      for (i = 0; i < 5; i++)
12          y[i] = x[i];// 배열의 원소끼리 1:1로 대입한다. (배열의 복사)
13
14      printf("y = ");
15      for (i = 0; i < 5; i++)
16          printf("%d ", y[i]);
17      printf("\n");
18
19      return 0;
20  }
```

x와 y는 크기와 데이터형이 같은 배열이다.

실행결과

```
y = 10 20 30 40 50
```

배열의 비교

- 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

예제 : 배열의 비교

```c
#include <stdio.h>

int main(void)
{
    int x[5] = { 10, 20, 30, 40, 50 };
    int y[5] = { 10, 20, 30, 40, 50 };
    int i;
    int is_equal;

    if (x == y)              // x와 y의 주소가 같은지 비교한다.
        printf("두 배열의 주소가 같습니다.\n");

    is_equal = 1;            // 배열의 내용이 같은지를 나타내는 변수
    for (i = 0; i < 5; i++) {
        if (x[i] != y[i]) {   // 배열의 원소끼리 비교한다.
            is_equal = 0;     // 서로 다른 원소가 있으면 더이상 비교할 필요가 없다.
            break;
        }
    }
    if (is_equal == 1)       // 모든 원소가 같으면 is_equal은 1이다.
        printf("두 배열의 내용이 같습니다.\n");

    return 0;
}
```

실행결과

```
두 배열의 내용이 같습니다.
```

배열의 탐색 (1/2)

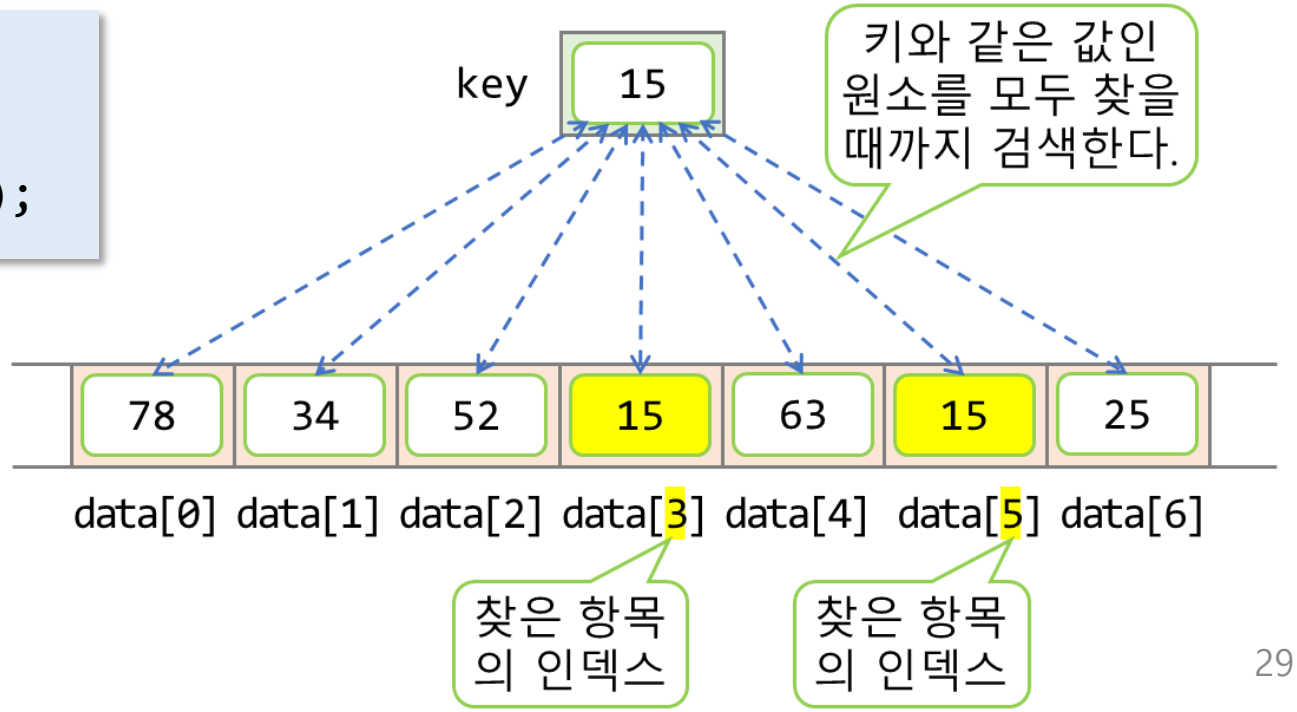
❖주어진 데이터 중에서 특정 값을 가진 항목을 찾는 기능

❖**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);
```



예제 : 배열의 탐색

```
03  int main(void)
04  {
05      int data[] = { 78, 34, 52, 15, 63, 15, 25 };
06      int size;
07      int key, i;
08
09      size = sizeof(data) / sizeof(data[0]);
10      printf("arr = ");
11      for (i = 0; i < size; i++)
12          printf("%d ", data[i]);
13      printf("\n");
14
15      printf("찾을 값(키)? ");
16      scanf("%d", &key);
17      for (i = 0; i < size; i++)
18          if (data[i] == key)   // 배열의 원소와 키 비교
19              printf("찾은 원소의 인덱스: %d\n", i);
20      return 0;
21  }
```

실행결과

```
arr = 78 34 52 15 63 15 25
찾을 값(키)? 15
찾은 원소의 인덱스: 3
찾은 원소의 인덱스: 5
```

key와 같은 값인 모든 원소의 인덱스를 출력한다.

배열의 탐색 (2/2)

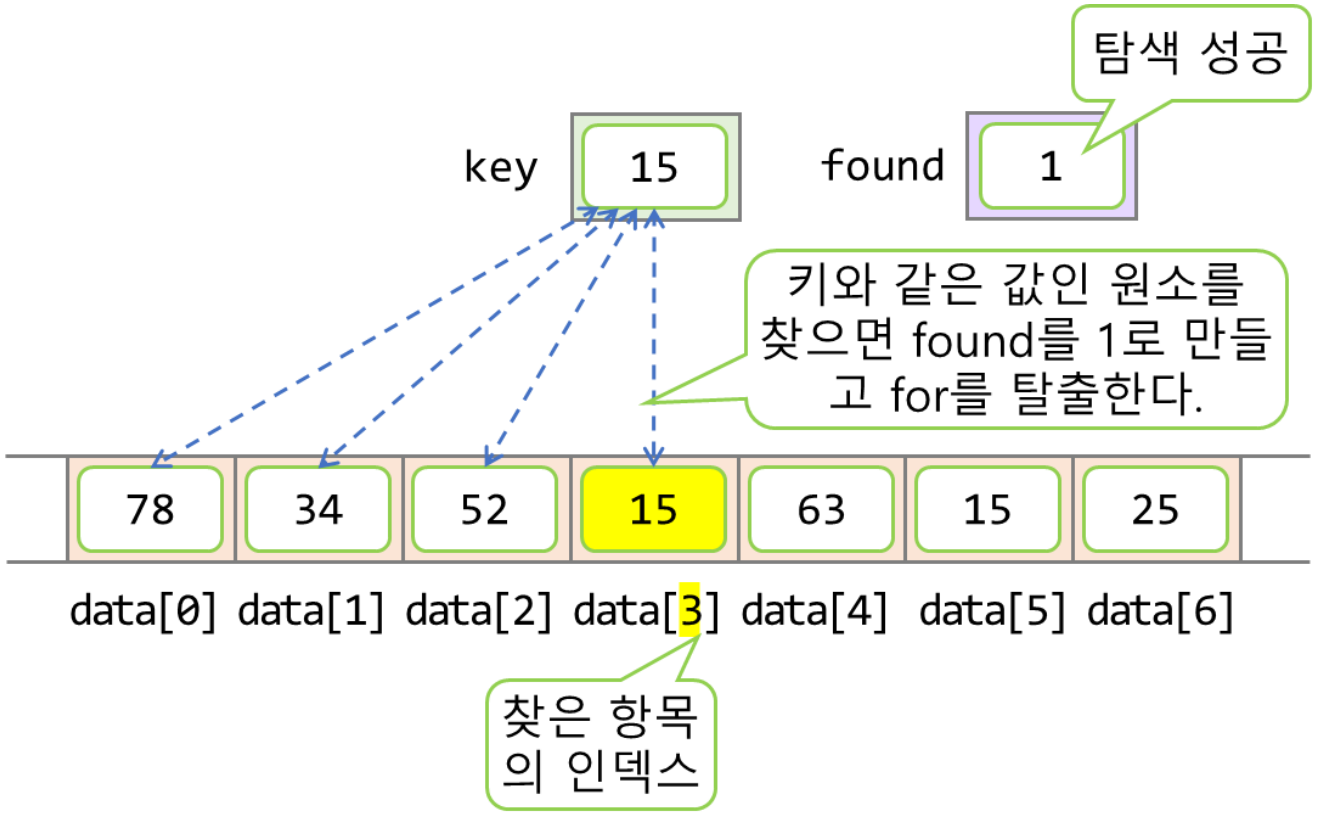
❖탐색키와 일치하는 첫 번째 항목만 찾는 경우

- 탐색키와 같은 값을 가진 원소를 찾으면 break로 for를 탈출한다

```
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



예제 : 탐색의 성공, 실패를 확인하는 경우

```
03  int main(void)
04  {
05      int data[] = { 78, 34, 52, 15, 63, 15, 25 };
06      int size;
07      int key, i;
08      int found;      // 탐색이 성공하면 1, 실패하면 0
09
10      size = sizeof(data) / sizeof(data[0]);
11      printf("arr = ");
12      for (i = 0; i < size; i++)
13          printf("%d ", data[i]);
14      printf("\n");
16      printf("찾을 값(키)? ");
17      scanf("%d", &key);
18      found = 0;
19      for (i = 0; i < size; i++) {
20          if (data[i] == key) {
21              found = 1;
22              break; // 탐색 성공 시 for 탈출
23          }
24      }
25      if (found == 1) // 탐색 성공인 경우 i가 찾은 항목의 인덱스이다.
26          printf("찾은 원소의 인덱스: %d\n", i);
27      else
28          printf("탐색 실패\n");
29      return 0;
30  }
```

실행결과

```
arr = 78 34 52 15 63 15 25
찾을 값(키)? 15
찾은 원소의 인덱스: 3
```

key와 같은 값인 첫 번째 원소의 인덱스만 출력한다.

배열의 정렬

❖주어진 데이터를 조건에 따라서 나열하는 기능
- 오름차순(ascending order) 정렬 : 크기가 커지는 순서로 나열
- 내림차순(descending order) 정렬 : 크기가 작아지는 순서로 나열

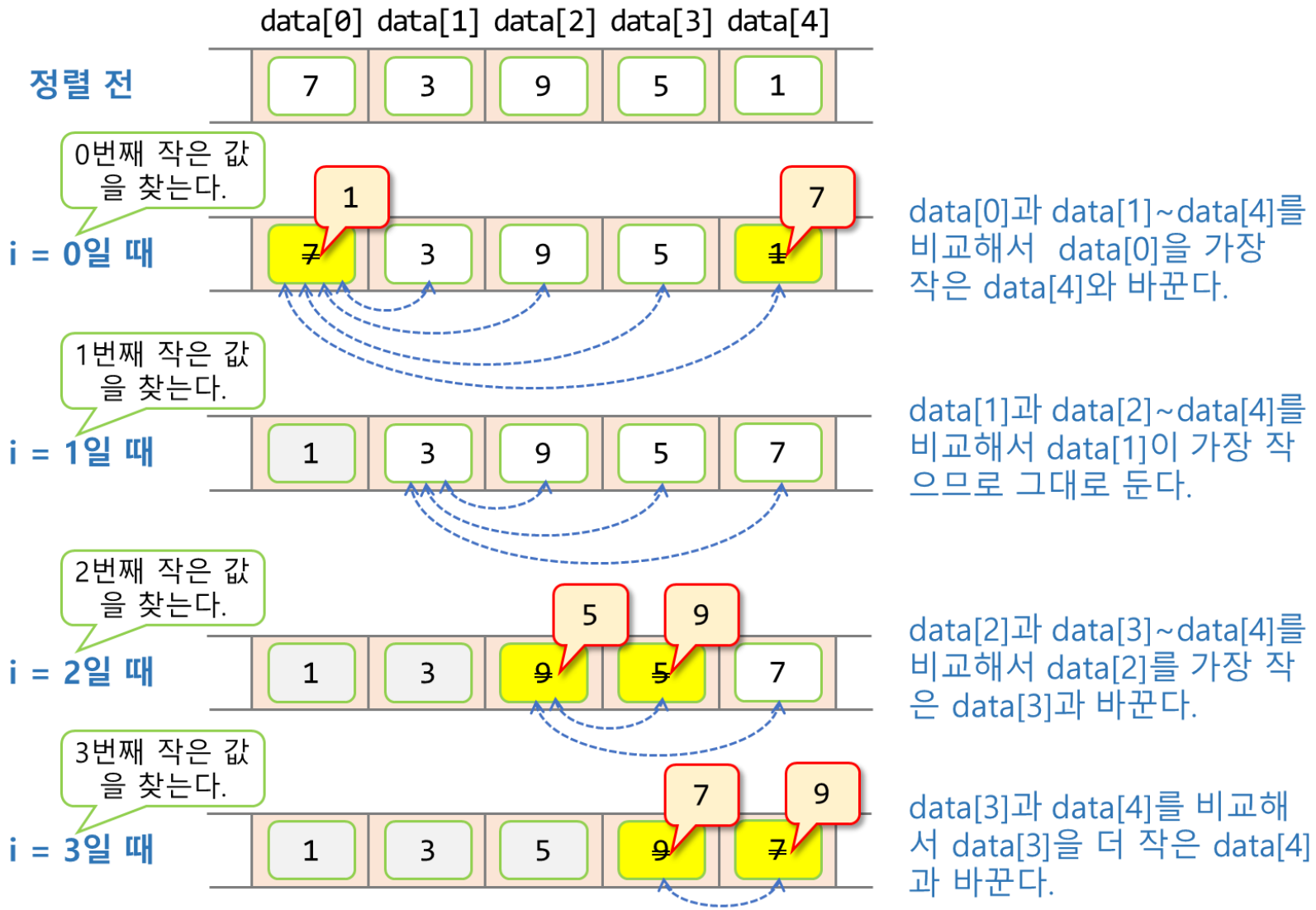
❖데이터가 정렬되어 있으면 데이터에 대한 여러 가지 작업이 간단해진다
- 최소값을 찾거나, 최대값을 찾는 작업은 배열의 0번 원소나 마지막 원소를 가져오면 된다
- 2진 탐색을 할 수 있으므로 빠른 탐색이 가능하다

❖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)

```
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] data[1] data[2] data[3] data[4]
정렬 전
7 3 9 5 1
0번째 작은 값을 찾는다.
i = 0일 때
1
7
~~7~~ 3 9 5 ~~1~~
data[0]과 data[1]~data[4]를 비교해서 data[0]을 가장 작은 data[4]와 바꾼다.
1번째 작은 값을 찾는다.
i = 1일 때
1 3 9 5 7
data[1]과 data[2]~data[4]를 비교해서 data[1]이 가장 작으므로 그대로 둔다.
2번째 작은 값을 찾는다.
i = 2일 때
5
9
1 3 ~~9~~ ~~5~~ 7
data[2]과 data[3]~data[4]를 비교해서 data[2]를 가장 작은 data[3]과 바꾼다.
3번째 작은 값을 찾는다.
i = 3일 때
7
9
1 3 5 ~~9~~ ~~7~~
data[3]과 data[4]를 비교해서 data[3]을 더 작은 data[4]과 바꾼다.

예제 : 선택 정렬

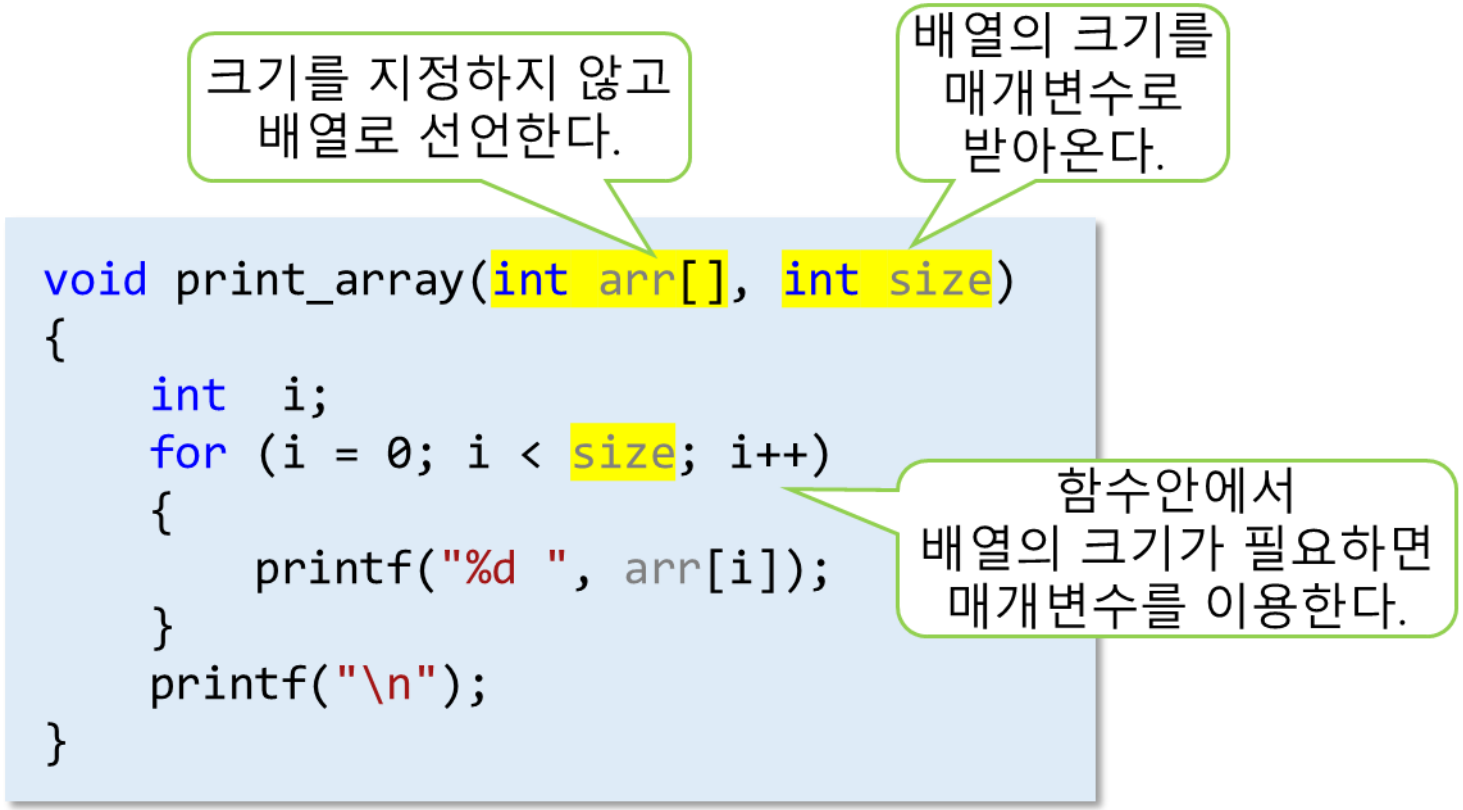
```
03  #define SIZE 5
04
05  int main(void)
06  {
07      int data[SIZE] = { 7, 3, 9, 5, 1 };
08      int i, j;
09      int index, temp;
10
11      for (i = 0; i < SIZE - 1; i++)      // 0~(i-1)까지는 정렬된 상태이다.
12      {
13          index = i;  // 정렬할 배열 중 가장 작은 원소의 인덱스
14          for (j = i + 1; j < SIZE; j++) {
15              // data[i]~data[SIZE-1]중 가장 작은 원소의 인덱스를 index에 저장
16              if (data[index] > data[j])   // 오름차순 정렬
17                  index = j;
18          }
19          // i번째 원소를 index에 있는 원소와 맞바꾼다.
20          if (i != index) {
21              temp = data[i];
22              data[i] = data[index];
23              data[index] = temp;
24          }   // i번째 원소가 i번째로 작은 값이 된다.
25      }
26      printf("정렬 후: ");
27      for (i = 0; i < SIZE; i++)
28          printf("%d ", data[i]);
29      printf("\n");
30      return 0;
31  }
```

실행결과

```
정렬 후: 1 3 5 7 9
```

함수의 인자로 배열 전달하기

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



예제 : print_array 함수의 정의 및 호출

```c
#include <stdio.h>
#define MAX 10
void print_array(int arr[], int size);     // 함수 선언

int main(void)
{
    int scores[] = { 99, 98, 67, 72, 90, 82 };
    int size = sizeof(scores) / sizeof(scores[0]);
    int arr[MAX] = { 0 };

    print_array(scores, size);          // 크기가 6인 int 배열 출력
    print_array(arr, MAX);              // 크기가 10인 int 배열 출력
    return 0;
}
void print_array(int arr[], int size)           // 배열의 원소를 출력하는 함수
{
    int  i;
    for (i = 0; i < size; i++)
        printf("%d ", arr[i]);
    printf("\n");
}
```

실행결과

```
99 98 67 72 90 82
0 0 0 0 0 0 0 0 0 0
```

예제 : copy_array 함수의 정의 및 호출

```
02  #define SIZE 7
03  void copy_array(int source[], int target[], int size);
04  void print_array(int arr[], int size);
05
06  int main(void)
07  {
08      int x[SIZE] = { 10, 20, 30, 40, 50 };
09      int y[SIZE] = { 0 };
10
11      printf("x = ");
12      print_array(x, SIZE);
13      copy_array(x, y, 5);
14      printf("y = ");
15      print_array(y, SIZE);
16      return 0;
17  }
19  void copy_array(int source[], int target[], int size)
20  {
21      int i;
22      for (i = 0; i < size; i++)
23          target[i] = source[i];          // 배열의 원소를 복사한다.
24  }
25
26  void print_array(int arr[], int size)
27  {
28      int  i;
29      for (i = 0; i < size; i++)
30          printf("%d ", arr[i]);
31      printf("\n");
32  }
```

실행결과

```
x = 10 20 30 40 50 0 0
y = 10 20 30 40 50 0 0
```

배열의 전달 (1/2)

- 함수의 매개변수는 배열 원소에 대한 포인터형으로 선언한다

```
void print_array(int *arr);
void print_array(int arr[]);
```

- 함수를 정의할 때 배열의 크기가 필요하면 배열의 크기도 매개변수로 받아와야 한다

```
void print_array(int *arr, int size);
```

- 배열이 입력 매개변수일 때는 const 키워드를 지정한다

```
void print_array(const int *arr, int size);
```

배열의 전달 (2/2)

- 배열을 매개변수로 가진 함수를 호출할 때는 배열의 이름을 인자로 전달한다

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

- 함수를 정의할 때는 매개변수인 포인터를 배열 이름인 것처럼 인덱스와 함께 사용한다

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

다차원 배열의 개념 (1/3)

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

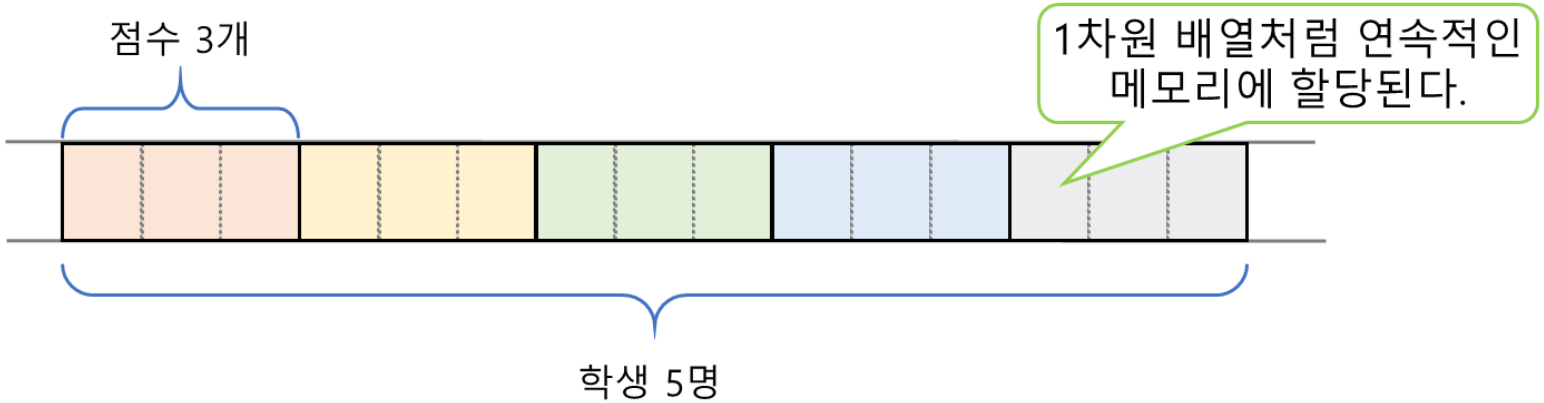
```
int scores[5][3];
```

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.

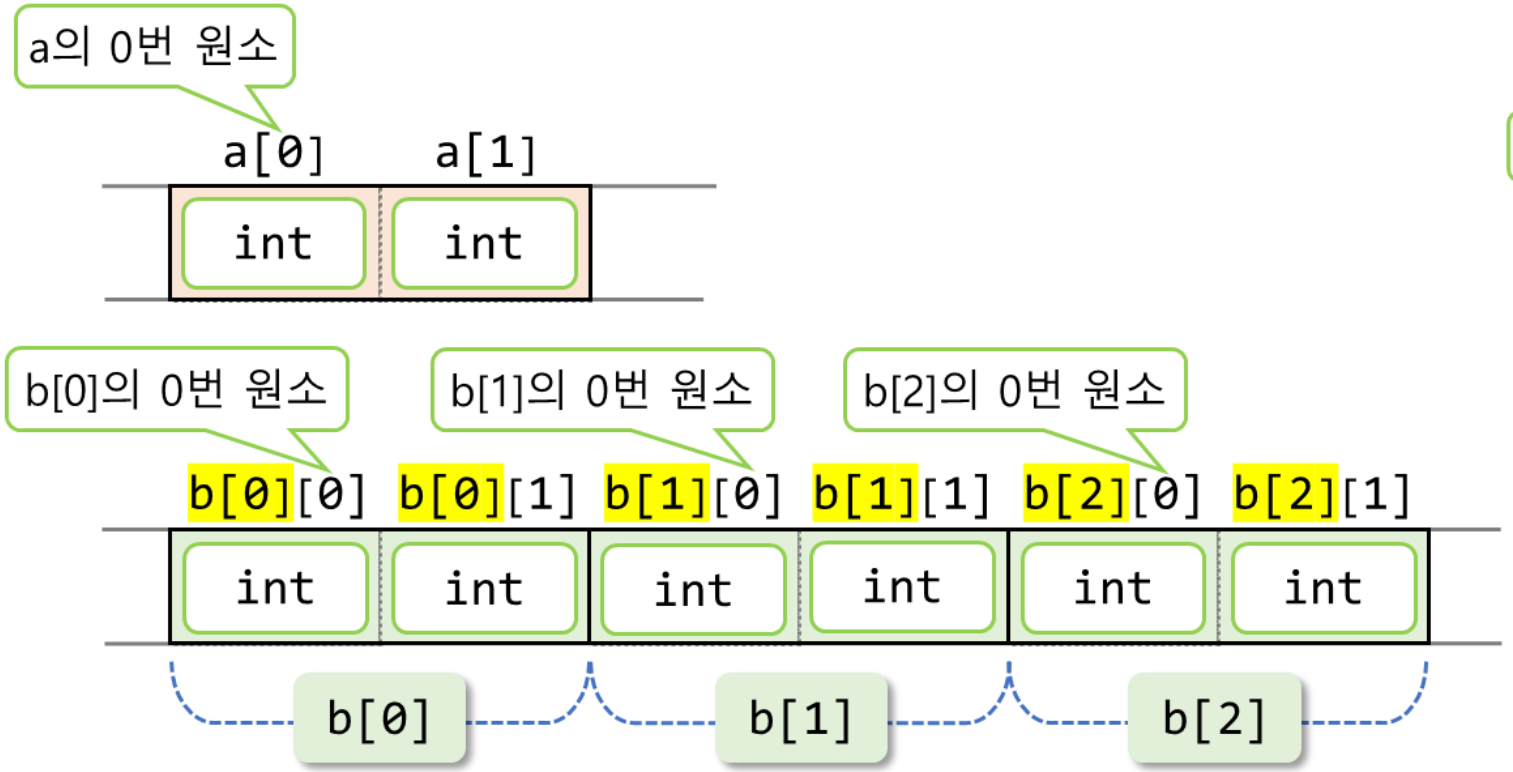


다차원 배열의 개념 (2/3)

```
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

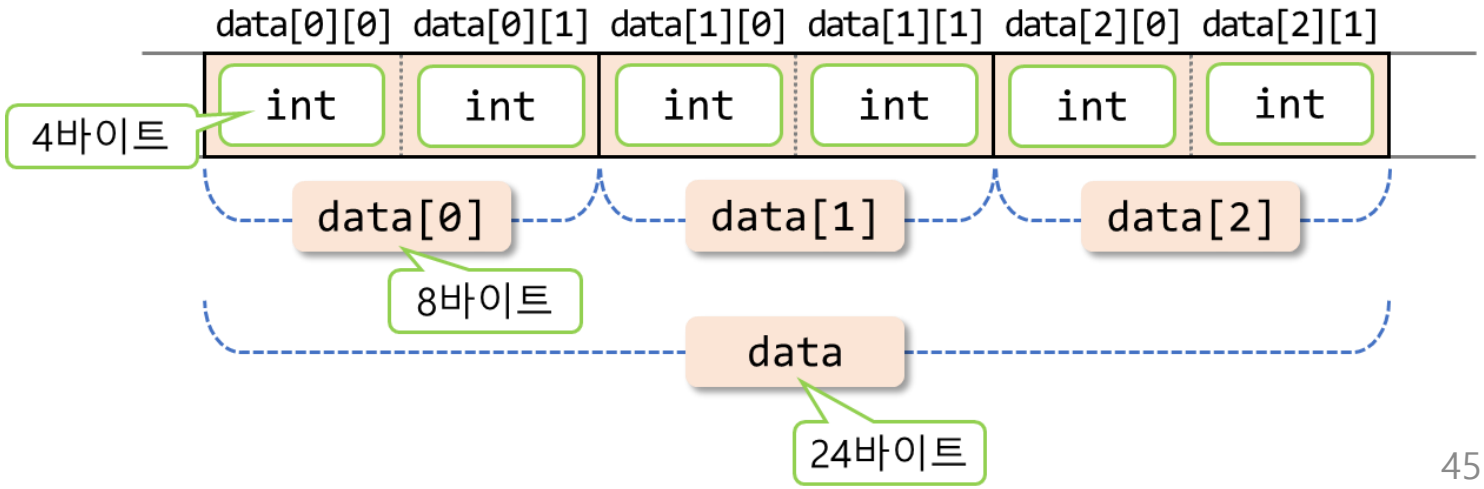
형식 데이터형 배열명[행크기][열크기];

사용예

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

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

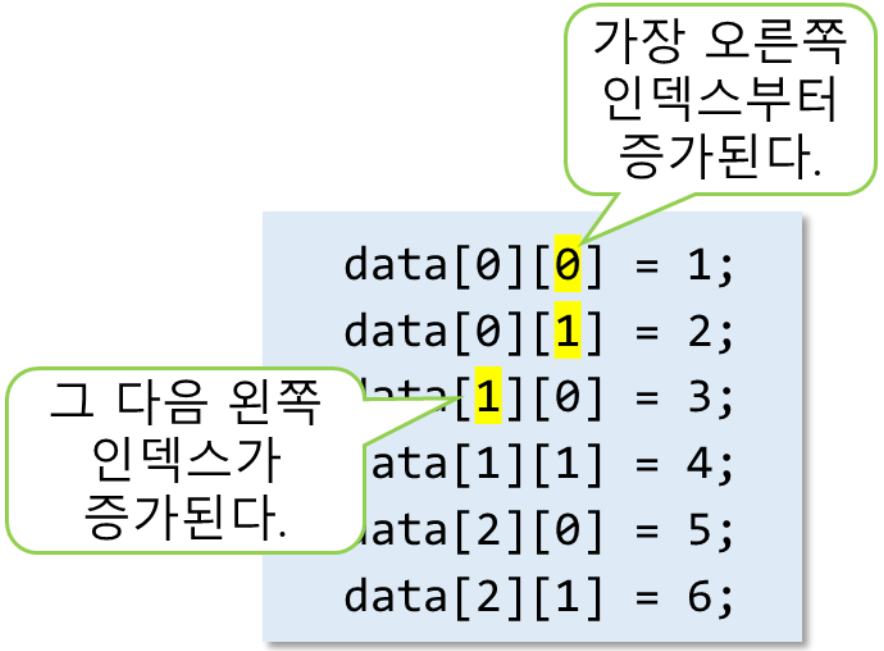
배열의 크기를 매크로 상수로 지정한다.

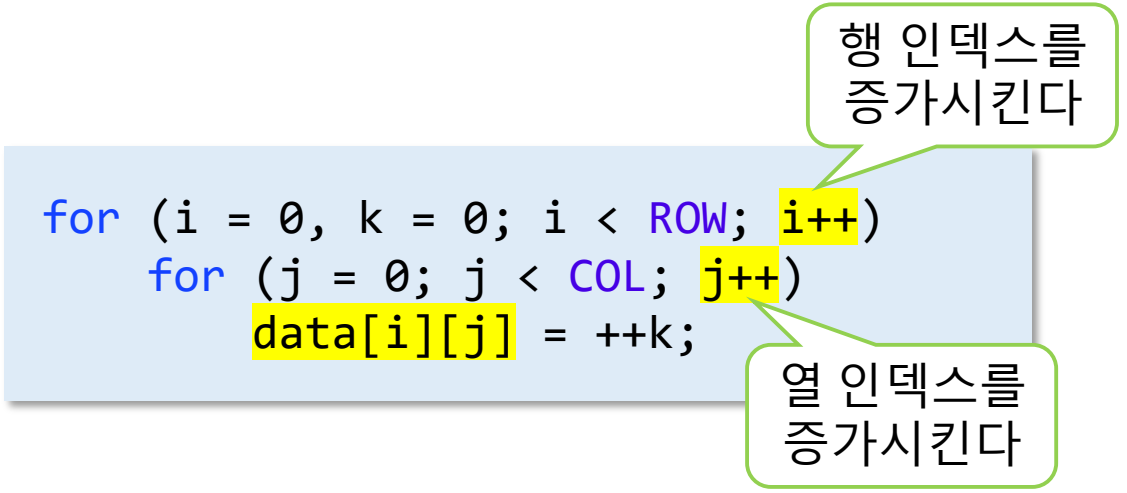


2차원 배열의 사용

❖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





예제 : 2차원 배열의 선언 및 사용

```c
#define ROW 3
#define COL 2

int main(void)
{
    int data[ROW][COL];
    int i, j, k;

    for (i = 0, k = 0; i < ROW; i++)    // 행 인덱스를 증가시킨다.
        for (j = 0; j < COL; j++)       // 열 인덱스를 증가시킨다.
            data[i][j] = ++k;           // 배열의 원소에 0부터 1씩 커지는 값을 저장한다.

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

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

실행결과

```
  1   2
  3   4
  5   6
sizeof(data) = 24
sizeof(data[0]) = 8
sizeof(data[0][0]) = 4
```

2차원 배열의 초기화

- 초기값을 열 크기의 개수만큼씩 { }로 묶어서 다시 { } 안에 나열한다
- 1차원 배열처럼 { } 안에 값만 나열할 수도 있다
- 초기값을 생략하면 나머지 원소를 0으로 초기화한다
- 초기값을 지정하는 경우에 배열의 행 크기를 생략할 수 있다

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

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

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

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

3

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

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

예제 : 2차원 배열의 초기화

```
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
```

추가 자료

2차원 배열 프로그램

```
int main()                                          /* 3명의 4과목 점수를 2차워 배열을 이용 저장
{                                                       총점, 평균을 구하고 출력하는 프로그램      */
    int i, j, tot, score[3][4];                     // 3명의 4과목 점수를 저장할 2차원 배열 score[3][4] 선언
    double avg;
    for(i=0; i<3; i++){                             // 3명이므로 3번 반복
        printf("네 과목의 점수를 입력하세요: ");
        for(j=0; j<4; j++)                          // 4과목이므로 4번 반복
            scanf("%d", &score[i][j]);              // 점수 입력
    }
    for(i=0; i<3; i++){
        tot=0;                                      // 각 학생의 점수를 새롭게 누적할 때마다 0으로 초기화
        for(j=0; j<4; j++)
            tot+=score[i][j];                       // 한 학생의 점수를 총점에 누적한다
        avg=tot/40;                                 // 한 명의 총점을 모두 누적한 후에 평균 계산
        printf("총점: %d, 평균: %2lf₩n", tot, avg);     // 총점, 평균출력
    }
}
```

질의 및 응답

참고문헌

- 천정아,『Core C Programming』, 연두에디션(2019)
- C가 보이는 그림책, ANK Co, Ltd , 성안당 (2018)
- Greg Perry, Dean Miller『어서와 C언어는 처음이지』, 천인국 옮김, 인피니티북스(2015)
- KELLEY (역 : 김명호 외),『A Book on C』, 홍릉과학출판사(2003)
- 윤성우,『 열혈 C 프로그래밍』, 오렌지미디어
- 천인국,『 쉽게 풀어쓴 C언어 Express』, 생능출판사
- 서현우,『 뇌를 자극하는 C 프로그래밍』, 한빛미디어
- 강성수,『 쾌도난마 C프로그래밍』, 북스홀릭
- 고응남,『 C프로그래밍 기초와 응용실습』, 정익사