```
int a = 10;
int *p1 = &a;
int **p2 = &p1;
```

변수 a의 값은 10
변수 p1의 값은 변수 a의 주소
변수 p2의 값은 변수 p1의 주소

변수 p2
타입 : int**
주소 100

변수 p1
타입 : int*
주소 108

변수 a
타입 : int
주소 116

p2의 값 : 108 번지
*p2의 값 : 116 번지
**p2의 값 : 10

p1의 값 : 116 번지
*p1의 값 : 10

a의 값 : 10

p2 : p2의 값 (포인터 변수의 값)
*p2 : p2가 참조하는 값(포인터가 가리키는 곳의 값)
**p2 : p2가 참조하는 포인터가 참조하는 값(포인터가 가리키는 주소가 가리키는 값)

p2의 타입은 : int**
p2의 타입은 : int
**p2의 타입은 : int

```
/*
    배열 중앙을 기준으로 오름 차순으로 정렬된 배열이 있습니다.
    배열의 값과 배열 경계 하한 값과 배열 경계 상한 값을 받아
    오름 차순으로 출력하는 함수를
*/

// TODO print 함수 선언 및 정의

int main() {
        int a[] = { 2, 4, 5, 7, 8, 1, 3, 4, 6, 9 };
        int b[] = { 4, 5, 6, 7, 2, 3, 4, 5, 6 };

        print(a, (int)(sizeof(a)/sizeof(a[0])));
        print(a, (int)(sizeof(b)/sizeof(b[0])));

        return 0;
}
```

실행결과

```
1 2 3 4 4 5 6 7 8 9
2 3 4 4 5 5 6 6 7
```

```c
/*
    첫 번째 파라미터로 전달받은 문자열을
    두 번째 파라미터로 받은 배열에
    최대 n-1 문자를 함수를 작성하세요.
*/

// TODO copy() 함수 선언 및 정의

int main() {
        char src[] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
        char dest[15] = { 0 };
        copy(src, dest, sizeof(dest));
        printf("%s\n", src);
        printf("%s\n", dest);

        return 0;
}
```

실행결과

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

```c
/*
    문자열을 입력받아 정수 값으로 변환하는 함수를 작성하세요.
*/

// TODO toInteger() 함수 선언 및 정의

int main() {
        char* str1 = "1234567890";
        char* str2 = "00001230";
        int number = toInteger(str1);
        printf("%d\n", number);
        number = toInteger(str2);
        printf("%d\n", number);

        getchar();
        return 0;
}
```

실행결과

```
1234567890
1230
```

배열의 주소는 배열의 시작 주소이며 배열의 첫 번째 원소의 시작 주소이다.

예) `int ar[] = { 1, 2, 3, 4, 5 };`

&ar	1차원 배열의 시작주소	(2차원 주소)
ar	1차원 배열의 주소	(1차원 주소)
&ar	1차원 배열의 첫 번째 원소의 시작 주소	(1차원 주소)
ar[0]	1차원 배열의 첫 번째 원소의 값	(0차원 값)

예) `int ar[][3] = { { 1, 2, 3 }, { 4, 5, 6 }, { 7, 8, 9 } };`

&ar	2차원 배열의 시작주소	(3차원 주소)
ar	2차원 배열의 주소	(2차원 주소)
&ar[0]	2차원 배열의 첫 번째 원소(행)의 시작 주소	(2차원 주소)
ar[0]	2차원 배열의 첫 번째 원소(행)의 주소	(1차원 주소)
&ar[0][0]	2차원 배열의 첫 번째 원소(행)의 첫 번째 원소(열)의 주소	(1차원 주소)
ar[0][0]	2차원 배열의 첫 번째 원소(행)의 첫 번째 원소(열)의 값	(0차원 값)

☞ 생각해볼 문제

`int* ar[3];`	&ar == ar 결과는?
`int (*ar)[3];`	&ar == ar 결과는?

- int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
- int* ptr = arr;

변수명	ptr	arr								
첨자		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]
주소	92	100 ptr + 0	104 ptr + 1	108 ptr + 2	112 ptr + 3	116 ptr + 4	120 ptr + 5	124 ptr + 6	128 ptr + 7	132 ptr + 9
값	100	1	2	3	4	5	6	7	8	9

- ptr은 int* 타입이므로 int형 배열의 주소가 저장될 수 있다.
- ptr에 저장된 값은 배열의 시작주소(100번지) 이다.
- *ptr은 *(ptr + 0)과 같으며 100번지에 저장된 값을 의미한다.
- ptr + 1은 ptr에 저장된 100번지로 부터 ptr이 가리키는 배열 원소 1개의 크기 만큼 이동한 위치를 의미한다.
- 그러므로 ptr + 1은 100 + 1 * sizeof(int)이므로 104번지가 된다.
- ptr + 1 연산에 의해 얻어진 104 번지는 1차원 주소이다.
- 값을 참조하기 위해 간접 참조 연산자를 사용하여야 한다.

- ptr + 1 -> 104번지, 배열의 두 번째 원소의 시작 주소
- *(ptr + 1) -> 104번지에 저장된 값은 2, 배열의 두 번째 원소의 값
- 포인터 표현식 *(ptr + 1) -> 배열 표현식 ptr[1]

- `int arr[][3] = { { 1, 2, 3 }`
  `, { 4, 5, 6 } };`
- `int (*ptr)[3] = arr;`

"포인터가 가리키는 곳의 데이터의 크기" 라는 말의 함정에 빠지지 말자.
정확히는 "포인터가 가리키는 대상체 타입의 크기"로 이해하도록 하자.

변수명	ptr	arr					
행첨자		[0] 타입은 int[3]			[1] 타입은 int[3]		
행주소		100 *(ptr+0)			112 *(ptr+1)		
열첨자		[0] int	[1] int	[2] int	[0] int	[1] int	[2] int
주소	92	100 *(ptr+0)+0	104 *(ptr+0)+1	108 *(ptr+0)+2	112 *(ptr+1)+0	116 *(ptr+1)+1	120 *(ptr+1)+2
값	100	1	2	3	4	5	6

- 포인터 변수 ptr은 하나의 데이터(원소)가 int[3] 인 배열 arr을 가리키는 포인터이다.
- 하나의 원소가 int[3]로 구성된 배열은 int[][3] 배열이다.(배열로 구성된 배열은 2차원 배열이다.)
- ptr은 2차원 배열 arr을 가리키고 arr은 int[3]으로 구성된 배열이므로
  ptr + 1 -> 100 + 1 * sizeof(int[3]) -> 112번지(arr의 두 번째 행의 시작 주소 : 2차원)
- *(ptr + 1)은 1차원 주소 112가 되며 이것은 ptr이 가리키는 2차원 배열 arr의 두 번째 행이 된다.
- *(ptr + 1) + 1은 *(ptr + 1)이 ptr이 가리키는 두 번째 행이 int형 3개로 구성된 배열이므로
  *(ptr + 1) + 1 -> 112 + 1 * sizeof(int) -> 116번지(arr의 두 번째 행의 두 번째 열의 주소: 1차원)
- 그러므로 *(*(ptr + 1) + 1)은 *(ptr + 1) + 1에 의해 구해진 주소 116번지가 가리키는 값 5가 된다.
- 이것은 포인터 표현식 *(*(ptr + 1) + 1)은 배열 표현식 ptr[1][1]과 동일하며 arr[1][1]이다.

- int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };
- int (*ptr)[9] = &arr;

변수명	ptr	arr									
첨자		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
주소	92	100	104	108	112	116	120	124	128	132	136
값	100	1	2	3	4	5	6	7	8	9	10

- ptr은 타입이 int(*)[9]이므로 1차원 배열 arr이 저장된 메모리 공간의 타입을 하나의 원소가 int[9]인 배열 타입으로 인식하게 된다.

변수명	ptr	arr									
행첨자		[0]									
열첨자		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
주소	92	100	104	108	112	116	120	124	128	132	136
값	100	1	2	3	4	5	6	7	8	9	10

- 그러므로 ptr의 값은 2차원 배열의 주소가 된다.
- *ptr은 ptr이 가리키는 2차원 배열의 첫 번째 행의 주소(1차원)가 되며
- **ptr은 ptr이 가리키는 2차원 배열의 첫 번째 행의 첫 번째 열의 값(0차원)이 된다.
- **ptr은 *(*(ptr + 0) + 0)과 같으며 배열 표현식으로 ptr[0][0]이 된다.
- 그렇다면 *(*(ptr + 1) + 0)은?

※ 다음 예제는 아래의 규칙을 위배 합니다.
포인터를 다른 타입으로 변환해서는 안된다.

- `int arr[] = { 1, 2, 3, 4, 5, 6, 7, 8, 9 };`
- `int (*ptr)[3] = &arr;`

변수명	ptr	arr									
첨자		[0]	[1]	[2]	[3]	[4]	[5]	[6]	[7]	[8]	[9]
주소	92	100	104	108	112	116	120	124	128	132	136
값	100	1	2	3	4	5	6	7	8	9	10

- ptr은 타입이 int(*)[3]이므로 1차원 배열 arr의 주소를 하나의 원소가 int[3]인 배열(2차원 배열) 타입으로 인식하게 된다.

변수명	ptr	arr								
행첨자		[0] int[3]			[1] int[3]			[2] int[3]		
열첨자		[0] int	[1] int	[2] int	[0] int	[1] int	[2] int	[0] int	[1] int	[2] int
주소	92	100	104	108	112	116	120	124	128	132
값	100	1	2	3	4	5	6	7	8	9

- ptr + 1은 100번지로부터 1 * sizeof(int[3]) 이동한 위치가 되므로 2차원 주소 112번지가 된다.
- *(ptr + 1)은 112번지가 가리키는 곳이며 데이터는 arr의 두 번째 행이다.
- *(ptr + 1) + 1은 *(ptr + 1)이 arr의 두 번째 행을 가리키고 두 번째 행은 int형 3개로 구성된 배열이므로
  112 + 1 * sizeof(int)만큼 떨어진 위치인 1차원 주소 116번지가 된다.
- *(*(ptr + 1) + 1)은 1차원 주소 116번지가 가리키는 곳의 값 5가 되며 배열 표현식 ptr[1][1]과 같다.

```
int* arr[3] = { NULL };
```

배열명	arr		
첨자	[0]	[0]	[0]
타입	int*	int*	int*
값	NULL	NULL	NULL

- 배열 arr의 원소 타입은 int* 타입이므로 arr 배열의 각 각의 원소는 0차원 변수의 시작주소 또는 1차원 int형 배열을 저장할 수 있다.

```
int a1[] = { 1, 2, 3 };
int a2[] = { 4, 5, 6 };
int a3[] = { 7, 8, 9 };
arr[0] = a1;
arr[1] = a2;
arr[2] = a3;
```

```
int arr[] = { 1, 2, 3 };
int* ptr = arr; 가 가능하므로

int* arr2[] = { NULL, NULL, NULL };
int** ptr2 = arr2; 도 가능하다.
```

배열 arr은 하나의 원소가 int* 타입 배열이므로 int** 타입 포인터에 저장할 수 있다.

```
int** ptr = arr;
```

배열명	arr int*[]		
첨자	[0]	[1]	[2]
타입	int*	int*	int*
값	100	88	76

arr[0]은 a배열을 가리킨다.

arr[1]은 b배열을 가리킨다.

arr[2]는 c배열을 가리킨다.

배열명	a int[3]		
첨자	[0] int	[1] int	[2] int
주소	100	104	108
값	1	2	3

배열명	b int[3]		
첨자	[0] int	[1] int	[2] int
주소	88	92	96
값	4	5	6

배열명	c int[3]		
첨자	[0] int	[1] int	[2] int
주소	76	80	84
값	7	8	9

배열명	arr int* []		
첨자	[0]	[1]	[2]
주소	184	192	200
타입	int*	int*	int*
값	100	88	76

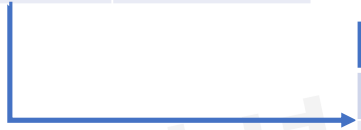
배열명	b int[3]		
첨자	[0] int	[1] int	[2] int
주소	88	92	96
값	4	5	6

※ 64bit 주소체계

- arr은 배열이고 시작주소는 184번지이다.
- arr + 1은 arr의 구성요소가 int* 형이므로 arr + 1 * sizeof(int*)이 되어 192번지(2차원주소)가 된다.
- *(arr + 1)은 배열표기식 arr[0]이 되며 192번지에 저장된 주소인 88번지, b 배열의 주소(1차원)가 된다.
- *(arr + 1) + 1은 배열 a2의 주소인 88번지로부터 1 * sizeof(int) 만큼 떨어진 주소 92번지(1차원)가 된다.
- *(*(arr + 1) + 1)은 92번지에 저장된 값 5가 되며 이것은 배열 표현식 arr[1][1]과 동일하다.
- 질문 sizeof(arr)의 값과 sizeof(arr[0])의 값은?

※ 64bit 주소체계

배열명	arr int* []		
첨자	[0]	[1]	[2]
주소	184	192	200
타입	int*	int*	int*
값	100	88	76



배열명	b int[3]		
첨자	[0] int	[1] int	[2] int
주소	88	92	96
값	4	5	6