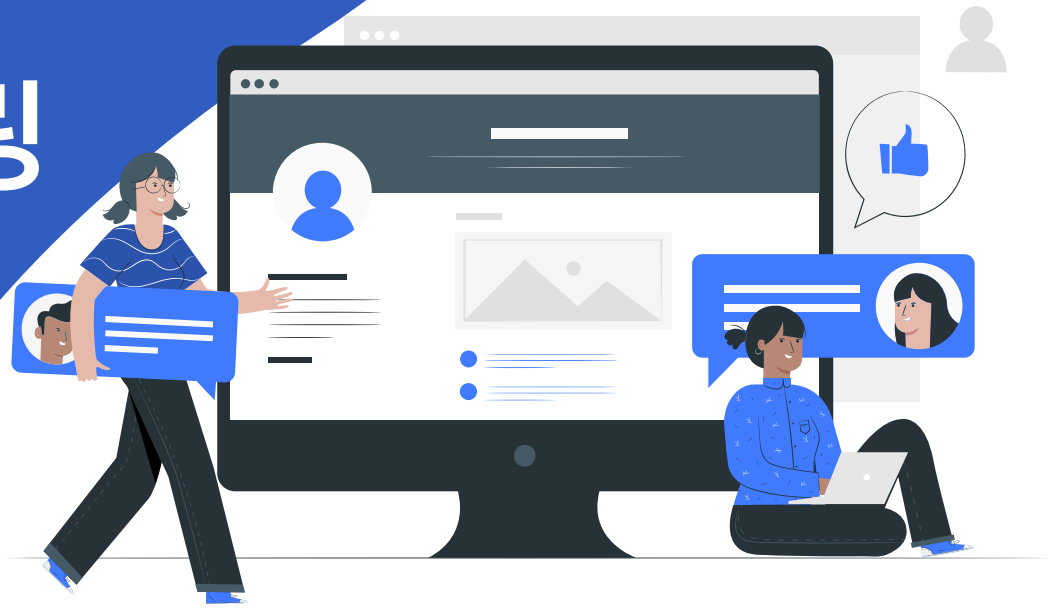
6주차 튜터링



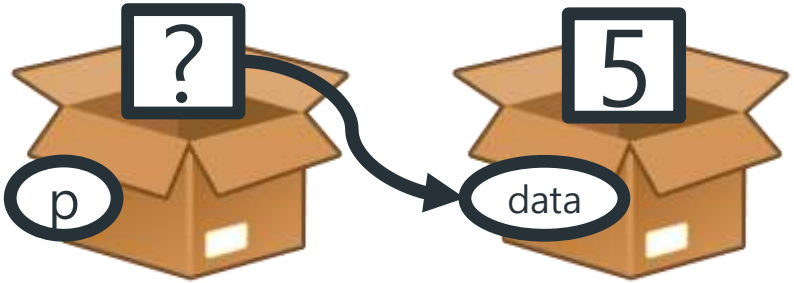
다차원 포인터

목차

다차원 포인터란?

- 우리가 지금까지 사용한 포인터는 1차원 포인터이다.

```
int *p, data = 5;
p = &data;
```



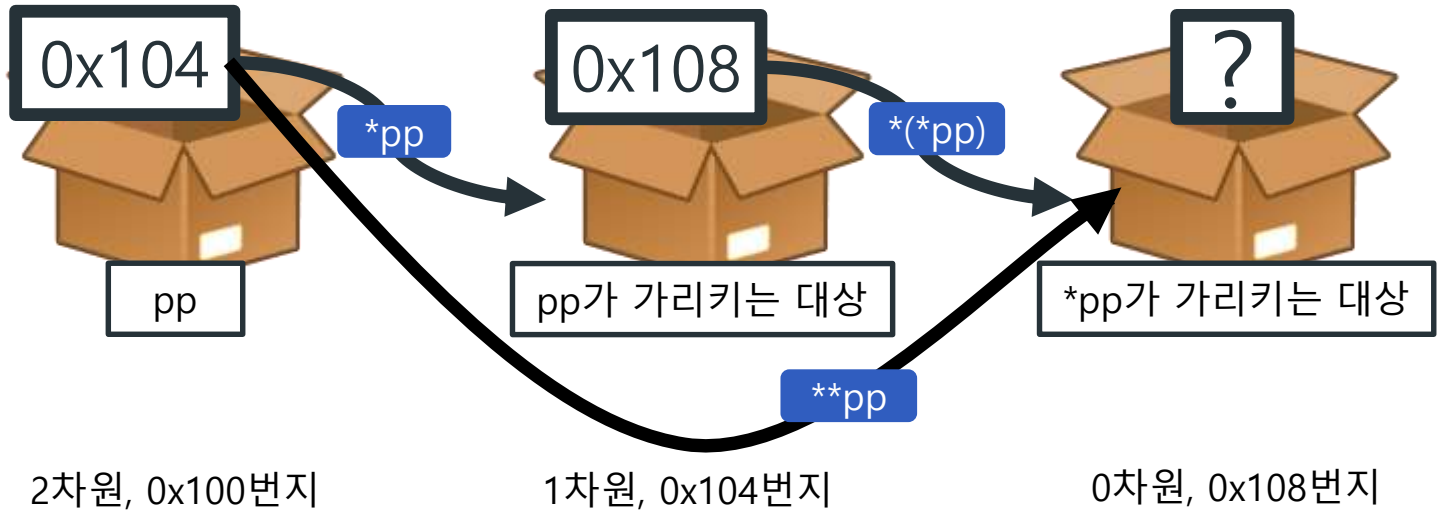
- 그럼 **다차원 포인터는 무엇이냐?**
  말 그대로 **여러 개의 포인터를 사용한 것**이다-!

```
int *p1;     // 1차원 포인터 : p1, *p1
int **p2;    // 2차원 포인터 : p2, *p2, **p2
int ***p3;   // 3차원 포인터 : p3, *p3, **p3, ***p3
```

2차원 포인터의 선언과 사용

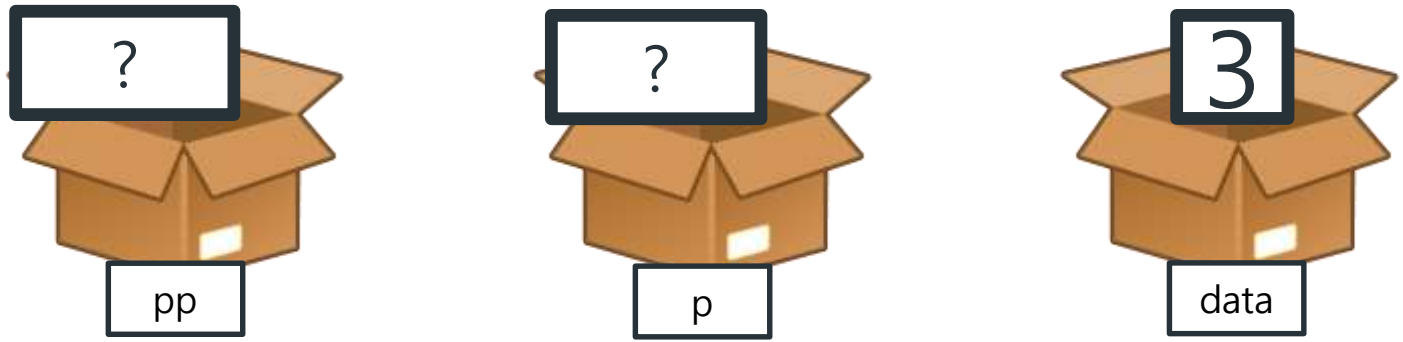
- 2차원 포인터는 주소 이동을 두 번 할 수 있다.

```
short **pp;
```



2차원 포인터의 선언과 사용

```
short **pp, *p, data = 3;
```



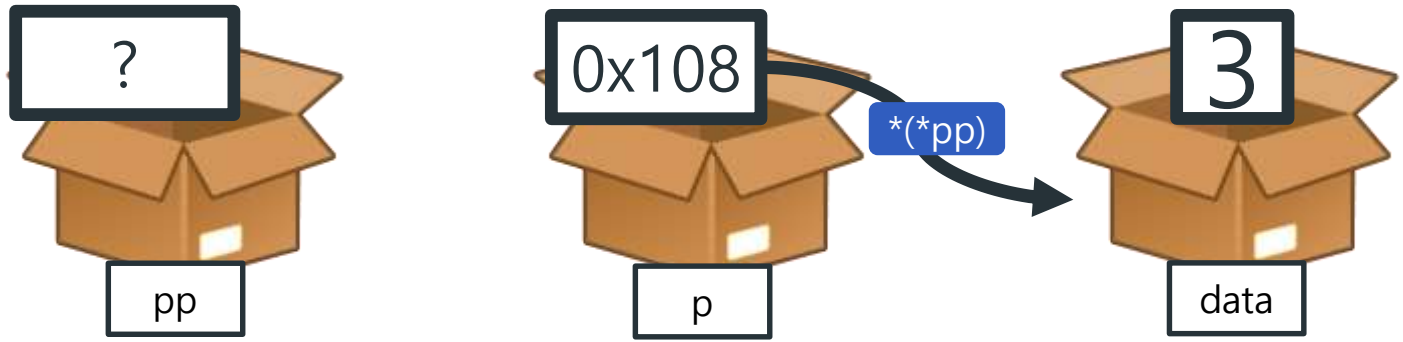
2차원, 0x100번지

1차원, 0x104번지

0차원, 0x108번지

2차원 포인터의 선언과 사용

```
short **pp, *p, data = 3;
p = &data;  // data 변수의 주소 값이 포인터 p에 저장됨.
```



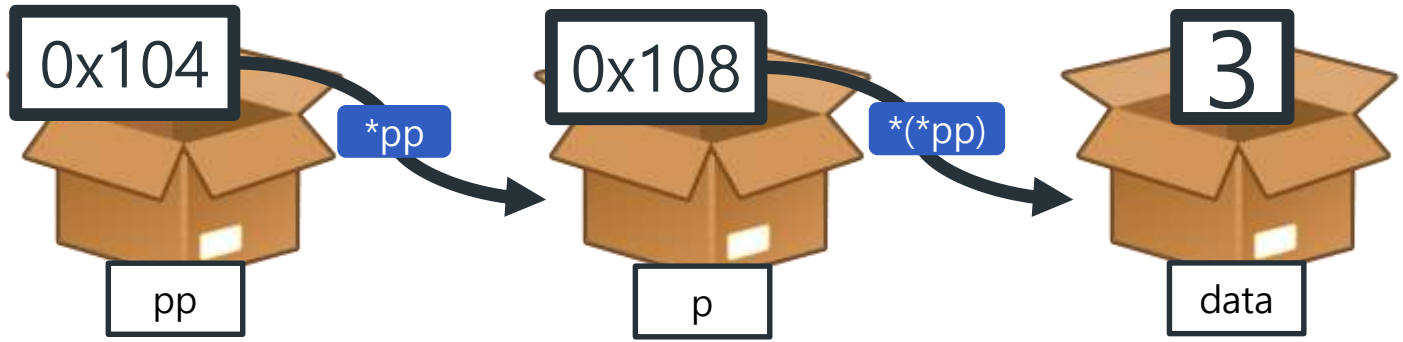
2차원, 0x100번지

1차원, 0x104번지

0차원, 0x108번지

2차원 포인터의 선언과 사용

```
short **pp, *p, data = 3;
p = &data;  // data 변수의 주소 값이 포인터 p에 저장됨.
pp = &p;    // 1차원 포인터 p의 주소 값이 2차원 포인터 pp에 저장됨
```



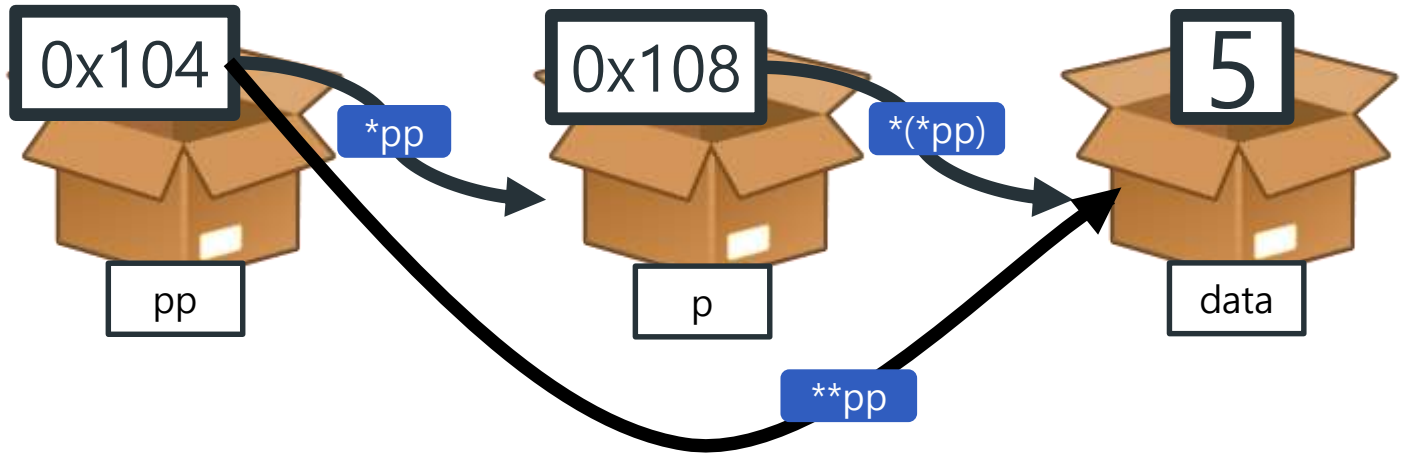
2차원, 0x100번지

1차원, 0x104번지

0차원, 0x108번지

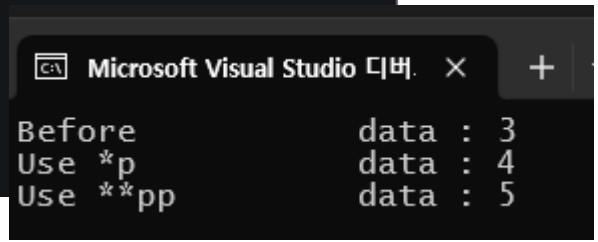
2차원 포인터의 선언과 사용

```
short **pp, *p, data = 3;
p = &data;  // data 변수의 주소 값이 포인터 p에 저장됨.
pp = &p;    // 1차원 포인터 p의 주소 값이 2차원 포인터 pp에 저장됨
**pp = 5;   // data 변수의 값이 3에서 5로 변경됨
```



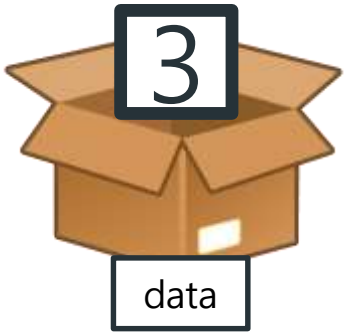
예제 코드 1

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main() {
    short data = 3;
    short* p = &data;    // data 변수의 주소 값을 1차원 포인터 p에 저장
    short** pp = &p;     // 1차원 포인터 p의 주소값을 2차원 포인터 pp에 저장

    printf("Before \t\t data : %d\n", data);
    *p = 4;
    printf("Use *p \t\t data : %d\n", data);
    **pp = 5;
    printf("Use **pp \t data : %d\n", data);

    return 0;
}
```



2차원 포인터가 가리키는 첫 대상이 일반 변수인 경우

```
short **pp, data = 3;
```



2차원, 0x100번지

0차원, 0x108번지

2차원 포인터가 가리키는 첫 대상이 일반 변수인 경우

```
short **pp, data = 3;
int my_ptr = (int)&data;          // my_ptr에는 정상적인 주소가 저장됨
```

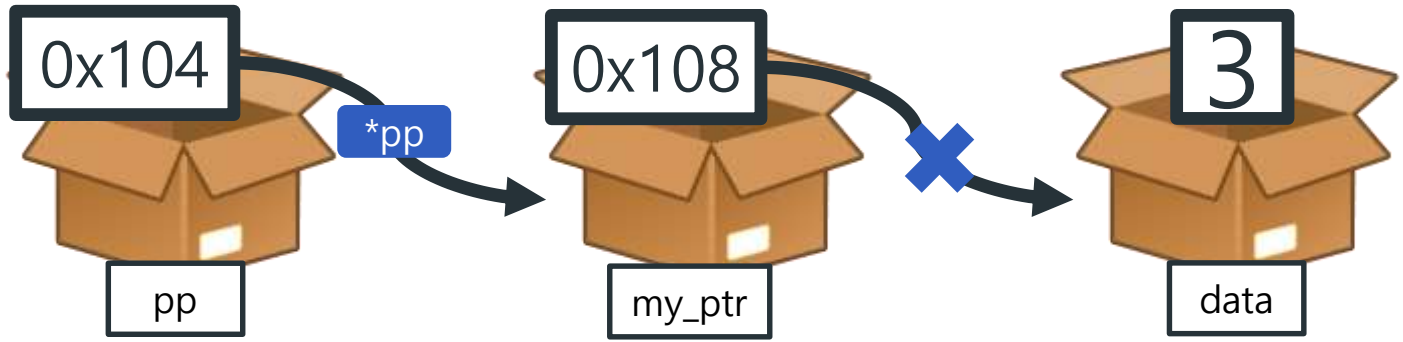
2차원, 0x100번지

0차원, 0x104번지

0차원, 0x108번지

2차원 포인터가 가리키는 첫 대상이 일반 변수인 경우

```
short **pp, data = 3;
int my_ptr = (int)&data;          // my_ptr에는 정상적인 주소가 저장됨
short **pp = (short **)&my_ptr; // my_ptr의 주소 값이 2차원 포인터 변수 pp에 저장됨.
```



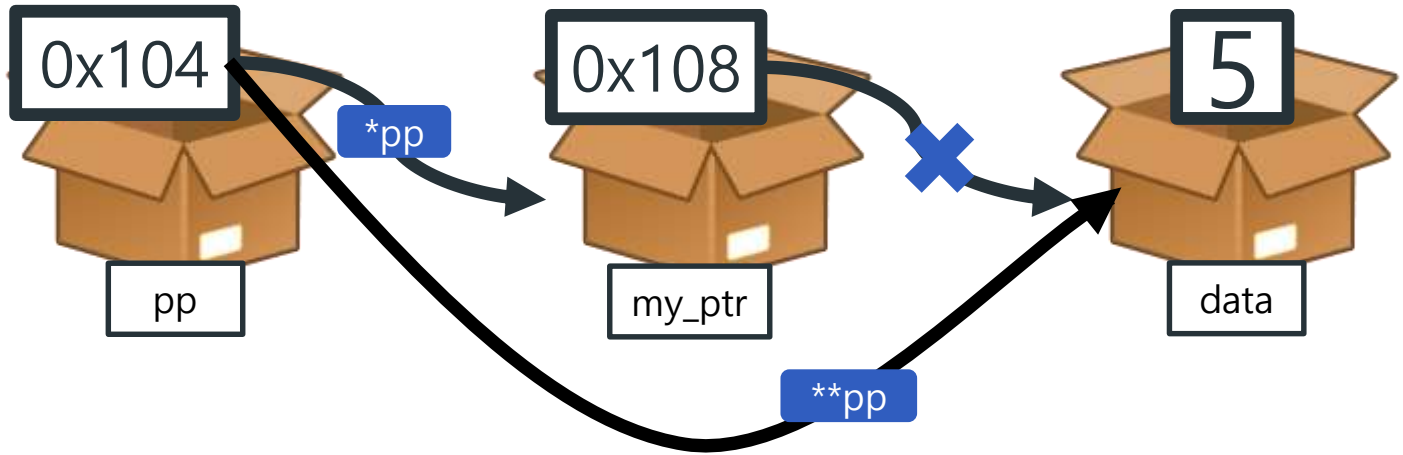
2차원, 0x100번지

0차원, 0x104번지

0차원, 0x108번지

2차원 포인터가 가리키는 첫 대상이 일반 변수인 경우

```
short **pp, data = 3;
int my_ptr = (int)&data;          // my_ptr에는 정상적인 주소가 저장됨
short **pp = (short **)&my_ptr; // my_ptr의 주소 값이 2차원 포인터 변수 pp에 저장됨.
**pp = 5;                         // data 변수의 값이 3에서 5로 변경됨
```



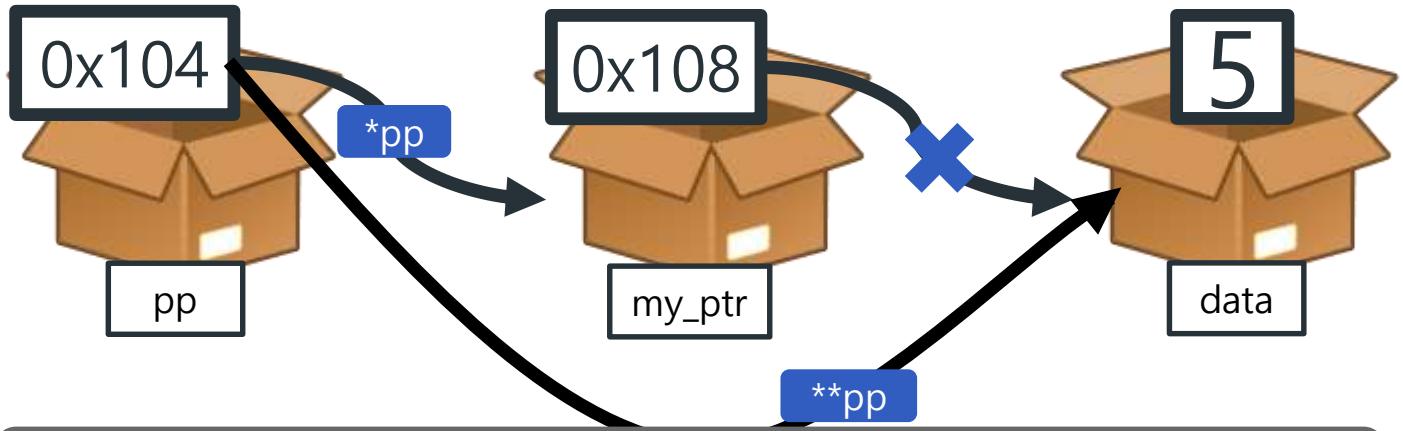
2차원, 0x100번지

0차원, 0x104번지

0차원, 0x108번지

2차원 포인터가 가리키는 첫 대상이 일반 변수인 경우

```
short **pp, data = 3;
int my_ptr = (int)&data;          // my_ptr에는 정상적인 주소가 저장됨
short **pp = (short **)&my_ptr; // my_ptr의 주소 값이 2차원 포인터 변수 pp에 저장됨.
**pp = 5;                         // data 변수의 값이 3에서 5로 변경됨
```



2차원 포인터 변수가 가진 주소값의 형식과 상관없이 4바이트 크기를 가지면
그 주소에 있는 변수의 값 변경이 가능하다.

2차원 포인터가 가리키는 첫 대상이 일반 변수인 경우

- 일반 변수 my_ptr는 *연산자를 사용할 수 없기 때문에 data 변수의 주소값을 저장하고 있더라도 이 주소로 이동할 수 없다.
- 하지만 2차원 포인터 변수인 pp는 **pp를 사용하여 두 번째 대상인 data 변수를 가리킬 때 첫 번째 대상에 저장된 주소값을 읽어서 data 변수를 가리킬 수 있다.
- 즉 첫 번째 대상인 my_ptr가 어떤 형식의 변수이든 상관없이 주소값만 정상적으로 저장되어 있다면 그 주소 값을 사용하여 두 번째 대상을 가리킬 수 있다.
- 세 번째 상자로 이동하여 값을 변경할 수 있다.

2차원 포인터와 동적 할당

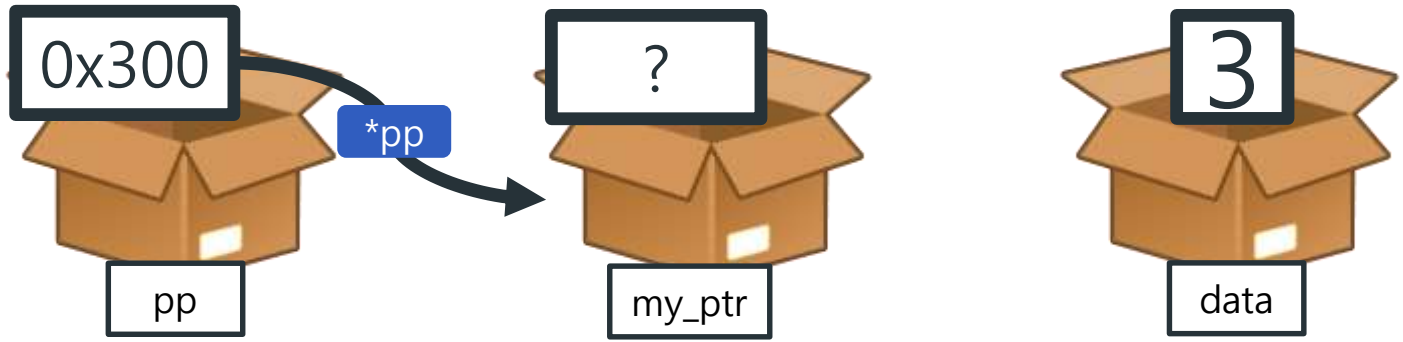
```
short **pp, data = 3;
```

2차원, 0x100번지

0차원, 0x108번지

2차원 포인터와 동적 할당

```
short **pp, data = 3;
pp = (short **)malloc(4);
```



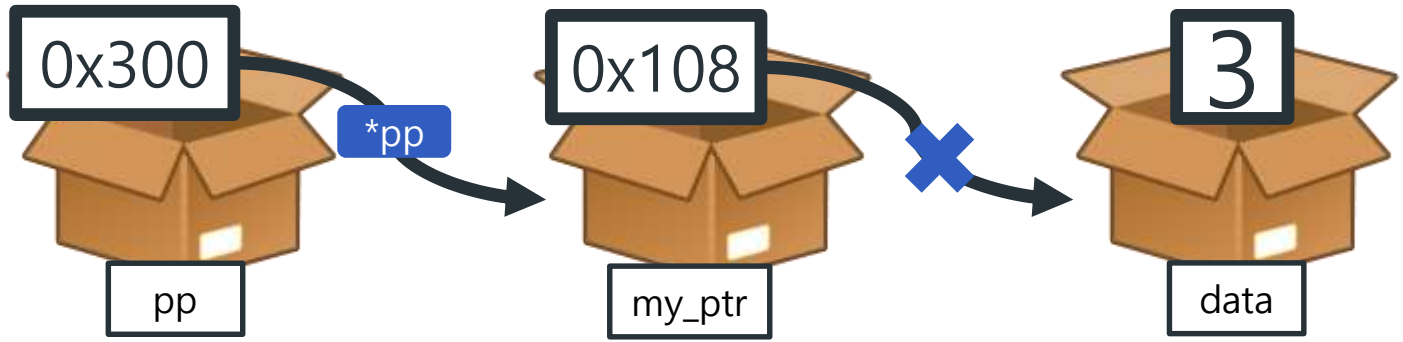
2차원, 0x100번지

Heap 영역, 0x300번지

0차원, 0x108번지

2차원 포인터와 동적 할당

```
short **pp, data = 3;
pp = (short **)malloc(4);
*pp = &data;     // data 변수의 주소값을 두 번째 상자에 저장
```



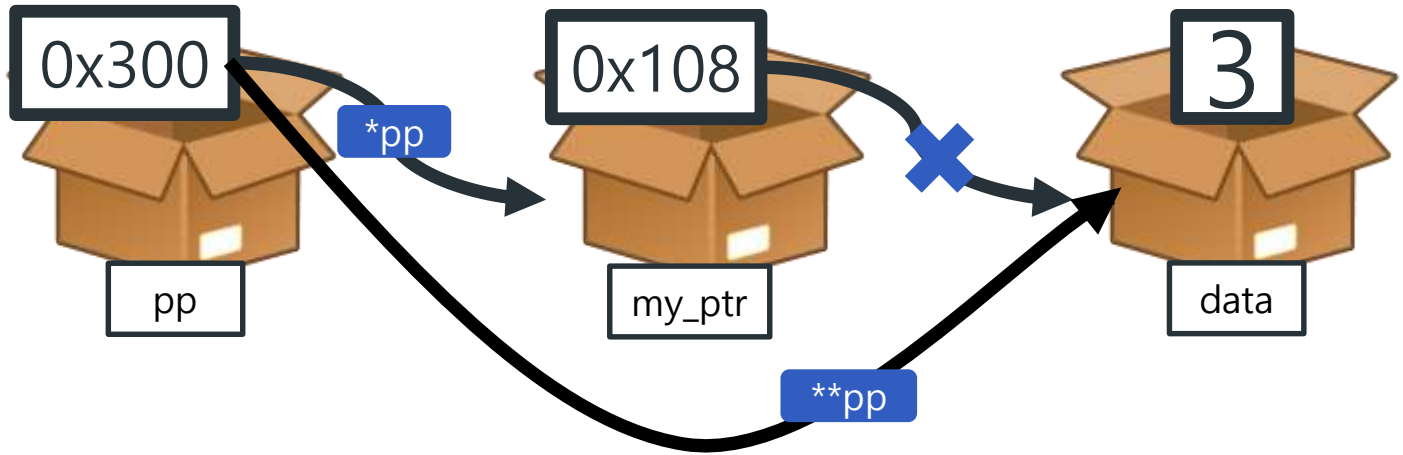
2차원, 0x100번지

Heap 영역, 0x300번지

0차원, 0x108번지

2차원 포인터와 동적 할당

```
short **pp, data = 3;
pp = (short **)malloc(4);
*pp = &data;     // data 변수의 주소값을 두 번째 상자에 저장
**pp = 5;        // data 변수의 값이 3에서 5로 변경
```



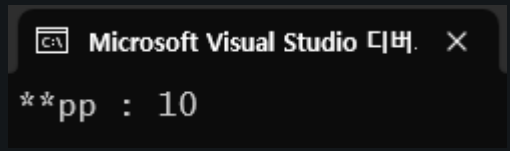
2차원, 0x100번지

Heap 영역, 0x300번지

0차원, 0x108번지

예제 코드 2

```c
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>

int main() {
    short** pp;
    pp = (short**)malloc(sizeof(short*)); // (short *)는 포인터이기 때문에 크기가 4바이트 이다.
    *pp = (short*)malloc(sizeof(short));

    **pp = 10;
    printf("**pp : %d\n", **pp);
    free(*pp);
    free(pp);

    return 0;
}
```



END