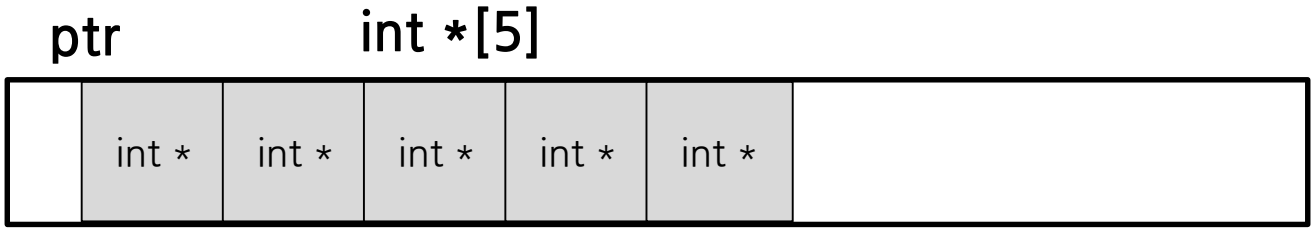
3차시 - 포인터(3)

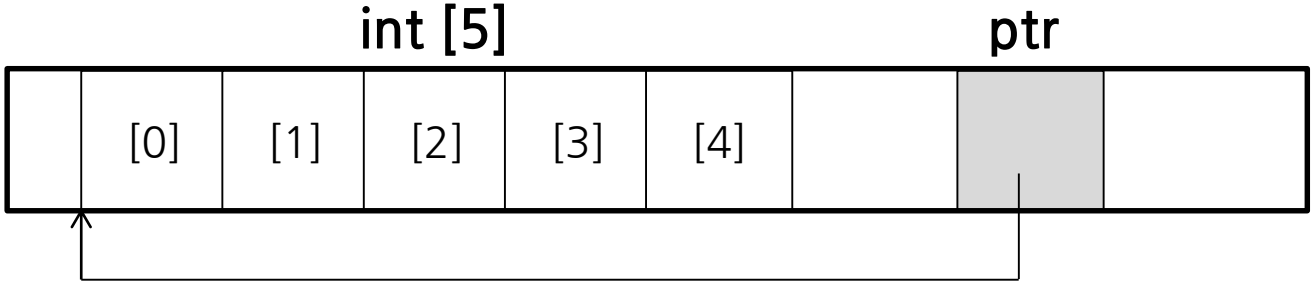
복잡한 포인터의 선언

- **int *ptr[5] ; // ① 5개짜리 배열 ② int * 포인터**
  - *ptr[0] = 10;

연산자 우선순위
[ ] > *

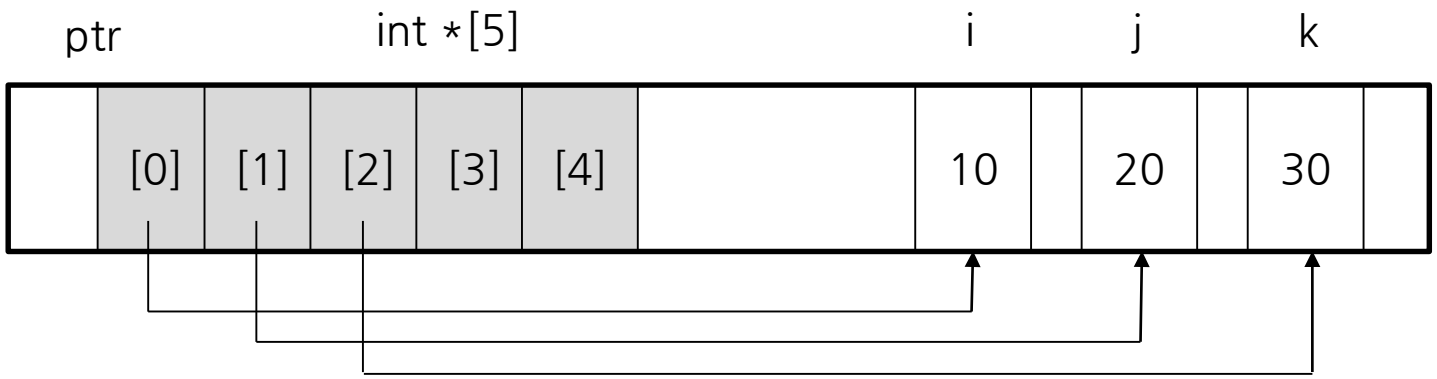


- **int (*ptr)[5]; // ① 포인터 ② int [5]형**
  - (*ptr)[0] = 10;



복잡한 포인터의 선언

- int *ptr[5] ; 의 사용 예

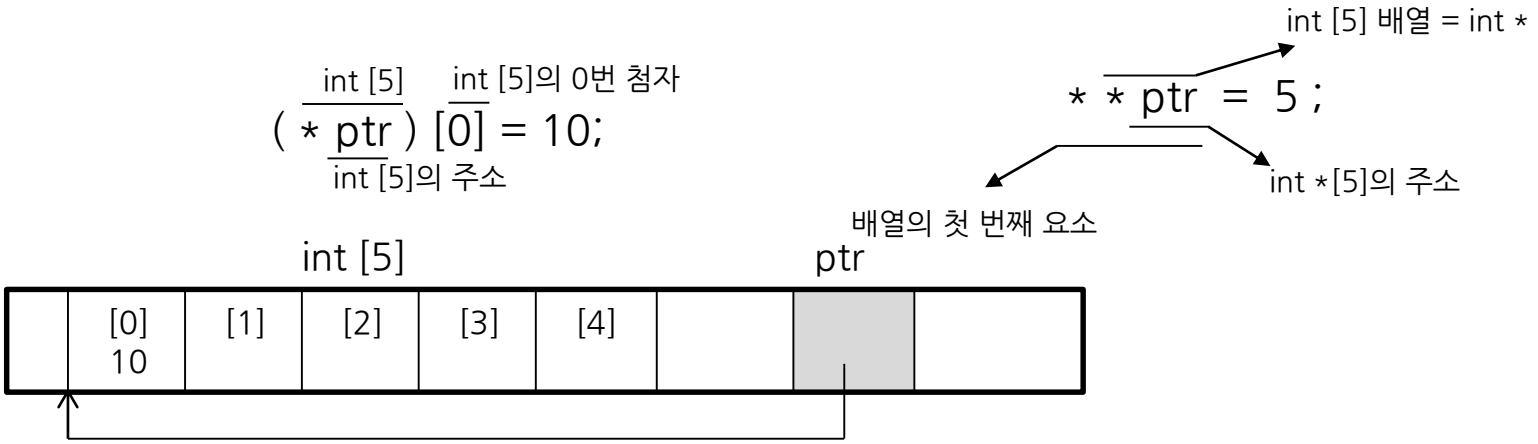



```
int     i;
int     *ptr[5];

ptr[0] = &i;
*ptr[0] = 10; // i = 10;
```

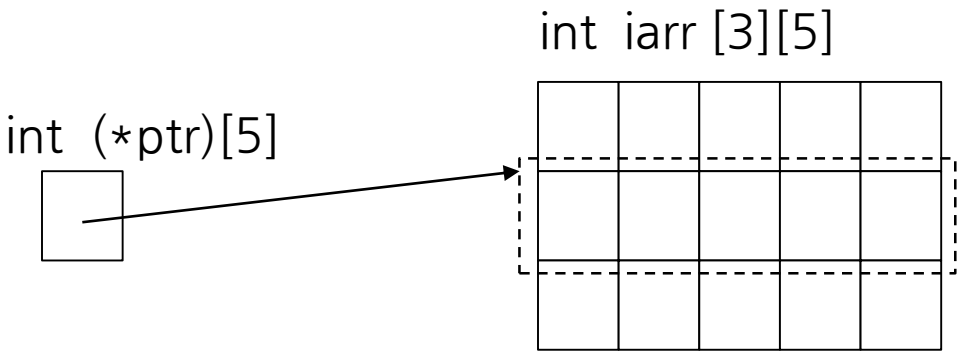
복잡한 포인터의 선언

- int (*ptr)[5] ; 의 사용 예

①



②

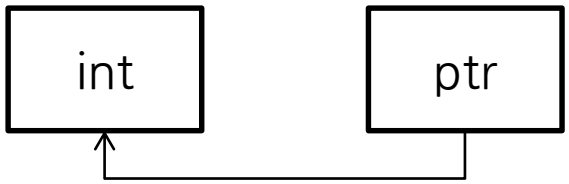


ptr = &iarr[1];
ptr = iarr + 1;

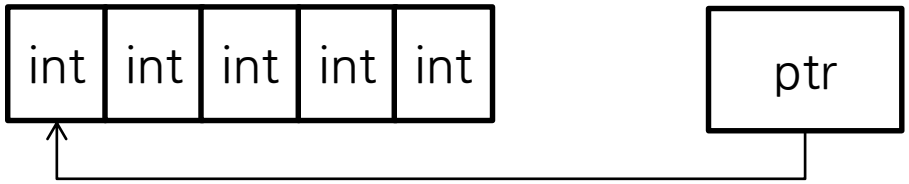
Quiz)

- 다음의 포인터 ptr 의 자료형은?

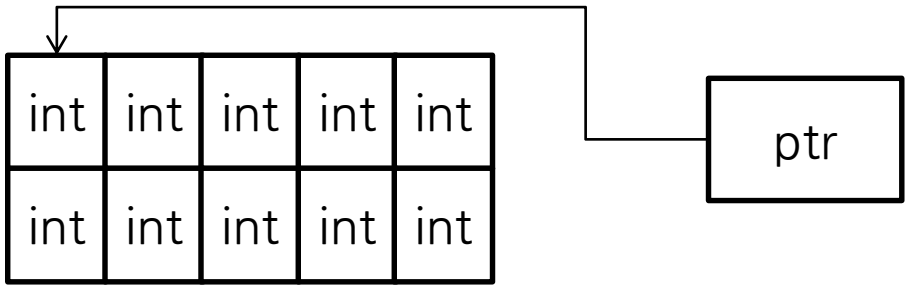
선언



① int *ptr



② int *ptr;
③ int (*ptr)[5]



④ int *ptr;
⑤ int (*ptr)[5]
⑥ int (*ptr)[2][5]

ptr++의 결과는 무엇일까?

Quiz)

```
int (*ptr)[5] = 1000;                    // ①

printf("%d\n", (ptr+1)- (ptr) );    // ②
printf("%p\n", ptr);
printf("%p\n", ptr+1);
```

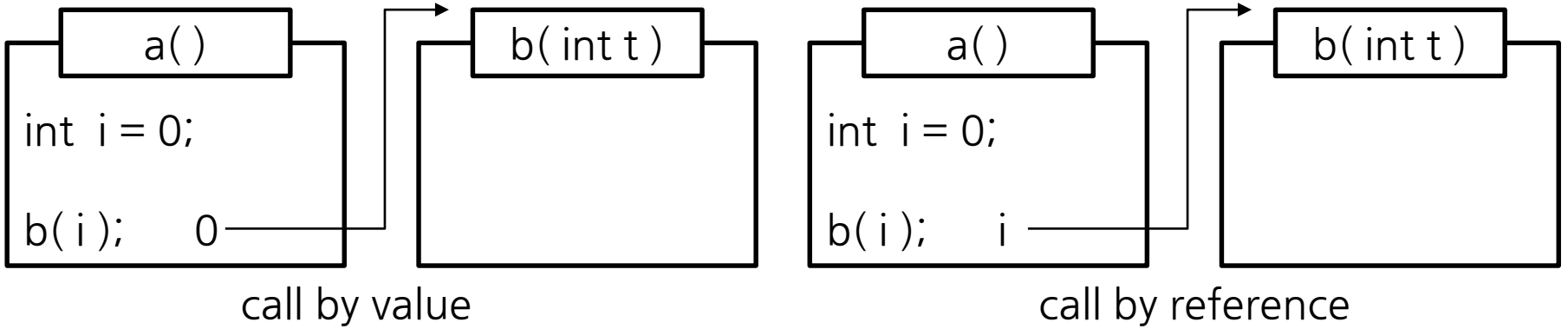
- ✓ ① 컴파일시 경고가 나왔다. 경고를 없애려면?
- ✓ ② 결과값은 1이 나왔다. (ptr+1)은 몇 번지일까?
- ✓ (ptr+1)- (ptr)의 결과가 1이 아니라, 실제 떨어진 주소만큼 나오게 하려면 어떻게 해야 할까?

우리가 알고 있는 함수의 인수 전달

- **함수를 호출할 때 인수는 값만 전달된다.**
  - func(a); // a 변수의 값만 func 함수에 전달된다.
- **배열은 인수 전달시 공유된다.**
  - func(array); // array 변수가 공유된다.

함수의 인수 전달

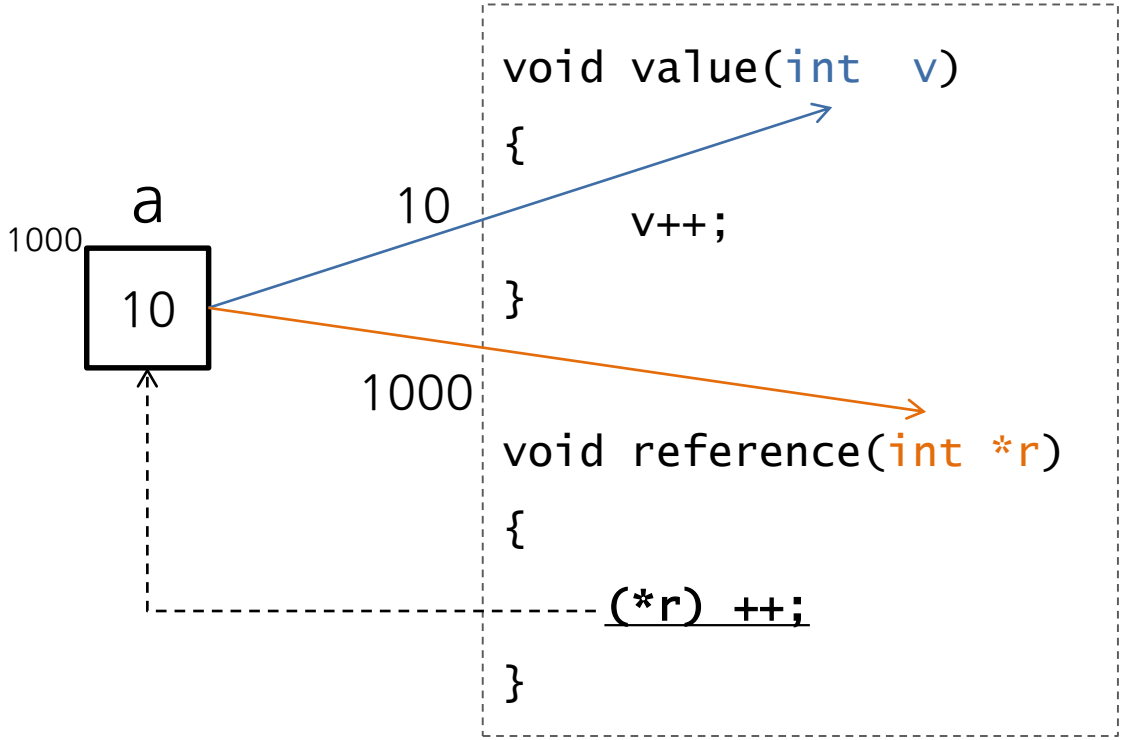
- **call by value vs call by reference**
  - by value : **변수의 값만** 전달한다. (C, Java, C++)
  - by reference : **변수 자체를** 전달한다. 하나의 변수가 공유된다. (Java, C++)



- **call by value로 call by reference의 효과**
  - 포인터를 전달한다. (하지만 call by value이다)

call by value로 reference의 효과

```
int main( )
{
    int  a = 10;
    value(a);
    printf("%d\n", a);
    reference(&a);
    printf("%d\n", a);
}
```



main 함수의 a 변수의 값을 바꿀 수 있는가 ?

포인터 전달의 효과

```
int main( )
{
    int  a = 10, b = 5;

    printf("a = %d, b = %d\n", a, b);
    swap(&a, &b);
    printf("a = %d, b = %d\n", a, b);
}
```

```
void  swap(int *i, int *j)
{
    int  temp;

    temp = *i;
    *i = *j;
    *j = temp;
}
```

- **swap 함수의 실행 결과 main 함수의 변수 a, b 의 값이 바뀐다.**
  - 함수의 실행 결과가 main 함수에 속한 변수에 영향을 준다.
    (이전까지는 불가능했던 일이 가능해진다.)
  - 어느 함수에 속한 변수든 포인터만 알면 swap 함수를 이용할 수 있다.
  - 함수의 기능이 강력해지면 더 많은 일을 할 수 있다.

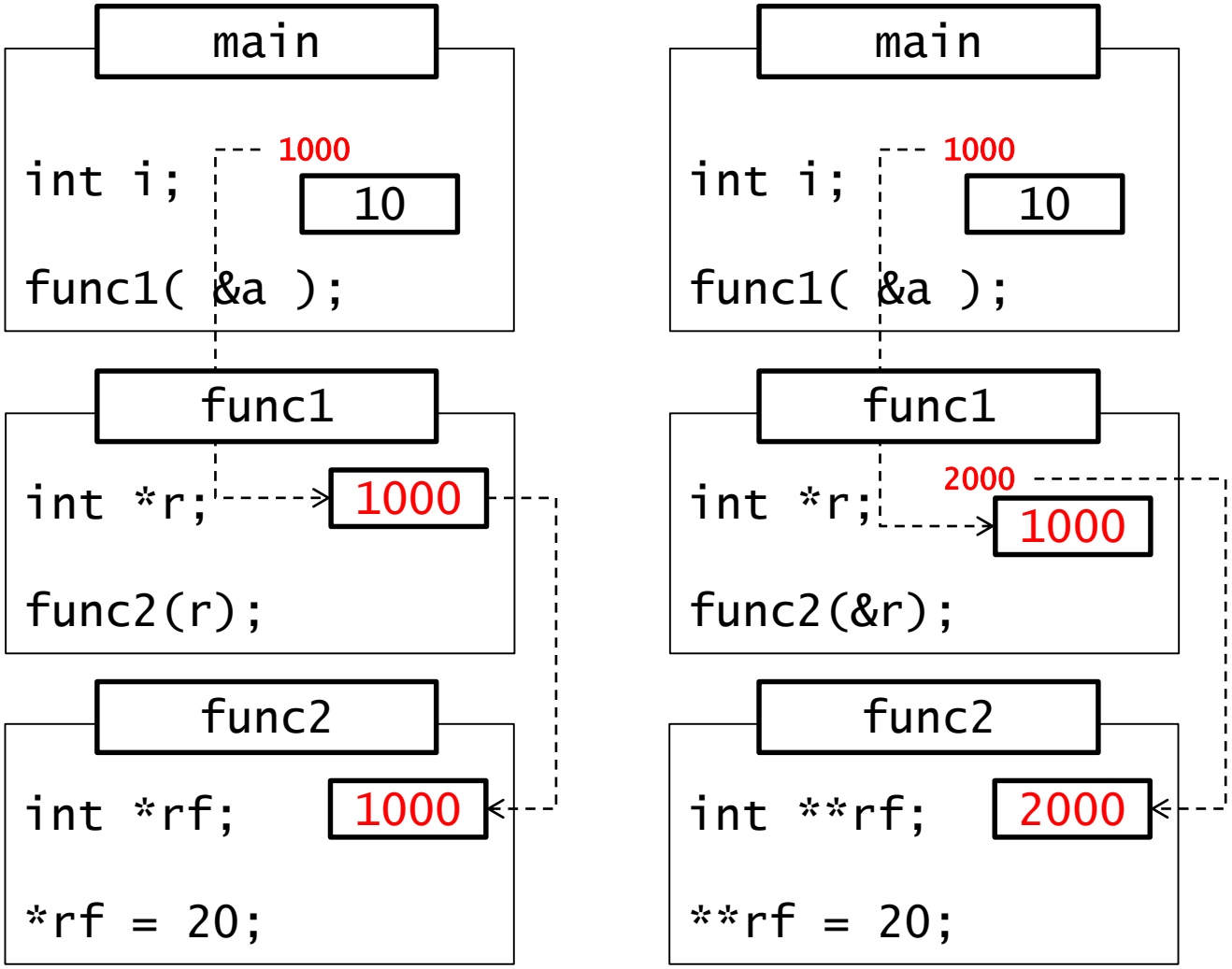
다수의 함수 호출

```c
#include <stdio.h>

void  func2(int *rf)
{
    *rf = 20; // 바로 main의 i에 접근
}

void  func1(int *r)
{
    func2(r);
}

int main( )
{
    int   i = 10;
    func1(&i);
    printf("%d\n", i );
}
```

```c
#include <stdio.h>

void  func2(int  **rf)
{
    **rf = 20; // 2번의 역참조를 통하여
}              // main의 i에 접근

void  func1(int *r)
{
    func2(&r);
}

int main( )
{
    int   i = 10;
    func1(&i);
    printf("%d\n", i );
}
```

포인터만 전달되면 어느 변수든 접근 가능하다.

다수의 함수 호출



정확히 이해하면 더욱 간단해진다.

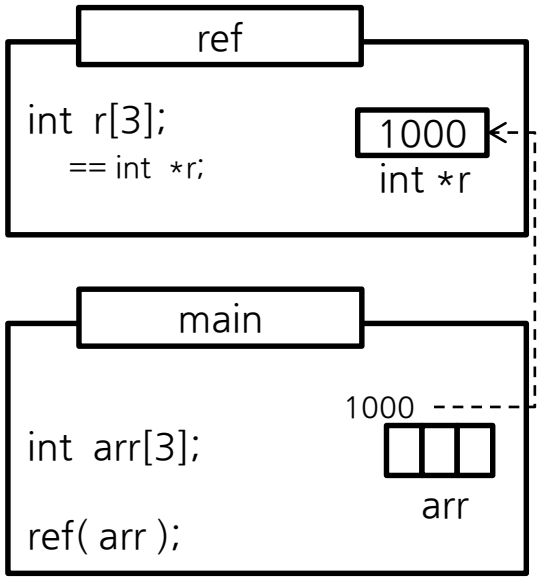
2차원 배열의 인수 전달

```
#include <stdio.h>
void  func(int mydata[4][3])
{
        printf("%d₩n", sizeof(mydata));
}

int  main(void)
{
        int     data[4][3] = { 3 };

        printf("%d₩n", sizeof(data));
        func(data);
}
```

- 결과는 무엇일까?
- 이유는?
  - 배열의 전달은 배열 전달이 아니다. 매개변수의 배열은 배열이 아니다.

배열의 전달

```
int main( )
{
    int  arr[3];

    ref (arr);
        // 배열의 이름은
        // 포인터 상수
        // 따라서 포인터가 전달
}
```




```
void ref( int r[3] ) {
    r[2] = 10 ;
    *(r+2) = 10;
}


void ref( int *r ) {
    r[2] = 10 ;
    *(r+2) = 10;
}
```

2차원 배열의 전달

```
int main( )
{
    int  arr[3][5];

// arr[2][2]에 10을 넣어라
    ref1 (arr);
    ref2 (arr[2]);
    ref3(&arr[2][2]);
}
```

arr[3][5]

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

```
void ref1(____r)
{
    ____= 10 ;
}

void ref2(_____r)
{
   _____= 10;
}

void ref3(_____r)
{
   _____= 10;
}
```

```
void ref1( int r[3][5] ) {
    r[2][2] = 10 ;
}
또는
void ref1( int (*r)[5] ) {
    r[2][2] = 10 ;
    (*(r+2))[2] = 10;
}

void ref2(int r[5] ) {
    r[2] = 10;
}
또는
void ref2(int *r ) {
    *(r+2) = 10;
}

void ref3(int *r )
{
    *r = 10;
}
```

2차원 배열의 전달

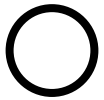
```
int main( )
{
    int  arr[3][5];

// arr[2][2]에 10을 넣어라
    ref1 (arr);
}
```

arr[3][5]

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

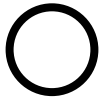
```
void ref1( int r[][5] ) {
    r[2][2] = 10 ;
}
```

배열의 앞 숫자는 생략 가능하다. 어차피 의미가 없다.



```
void ref1( int (*r)[5] ) {
    r[2][2] = 10 ;
    (*(r+2))[2] = 10;
}
```

배열의 뒤 숫자는 생략할 수 없다. 그 정보가 있어야 포인터를 계산할 수 있다.



```
void ref1( int **r ) {
    r[2][2] = 10 ;
}
```

에러는 없지만(경고는 발생) 포인터 위치를 계산할 수가 없다. r 의 규격을 알 수 없어 포인터의 위치 계산이 되지 않는다. 프로그램은 비정상 종료

Quiz)

```
int main( )
{
    int  arr[3][5];

// arr[2][2]에 10을 넣어라
    ref1 (arr);
}
```

arr[3][5]

1	2	3	4	5
6	7	8	9	10
11	12	13	14	15

경고를 무시하고 실행이 제대로 되도록 만들어라.

```
void ref1( int (*r)[5] ) {


}

void ref1(int **r )
{


}

void ref1(int *r ) {


}


void ref1(int  r ) {


}
```