

**-Part2-**

**제2장** 다차원 배열이란 무엇인가

# 학습목차

2.1 다차원 배열이란

2. 2 2차원 배열의 주소와 값의 참조

## 2.1 다차원 배열이란

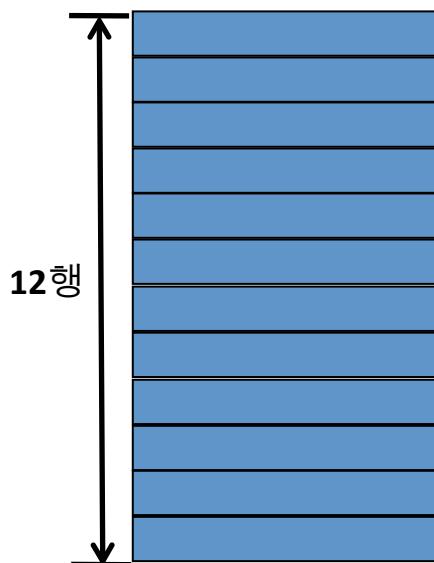
## 2.1 다차원 배열이란 (1/14)

- ▶ **다차원 배열:** 2차원 이상의 배열을 의미
- ▶ **1차원 배열과 다차원 배열의 비교**

1차원 배열

int array [12]

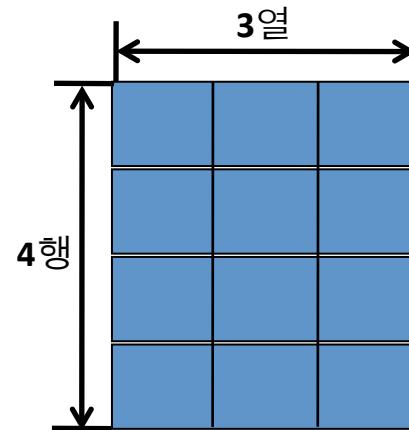
행



2차원 배열

int array [4][3]

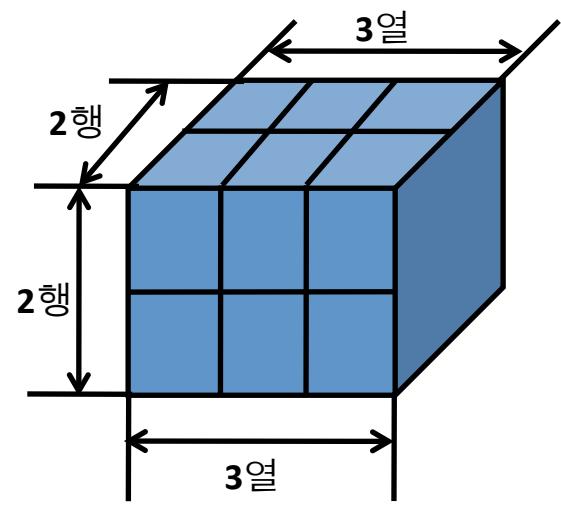
행 열



3차원 배열

int array [2][2][3]

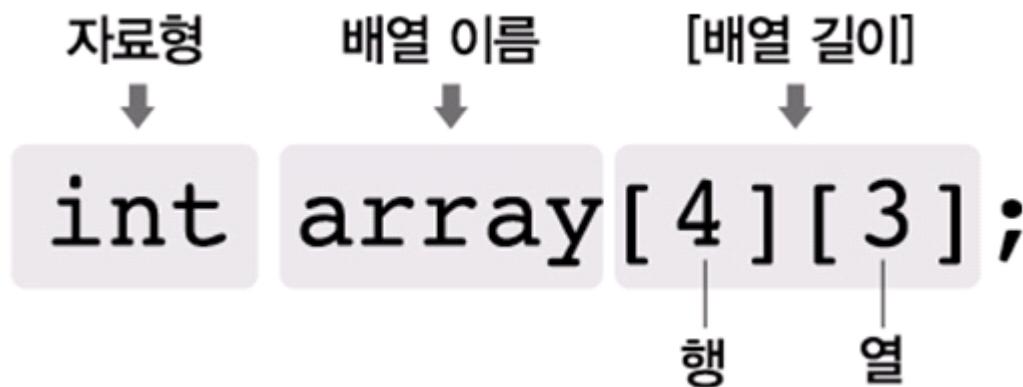
면 행 열



## 2.1 다차원 배열이란 (2/14)

### ▶ 2차원 배열의 선언

- ✓ 자료형: 배열의 자료형을 지정
- ✓ 배열 이름: 변수 이름과 마찬가지로 배열을 구분하는 배열의 이름
- ✓ 배열 길이: 배열 요소의 길이를 행(가로)과 열(세로)로 지정



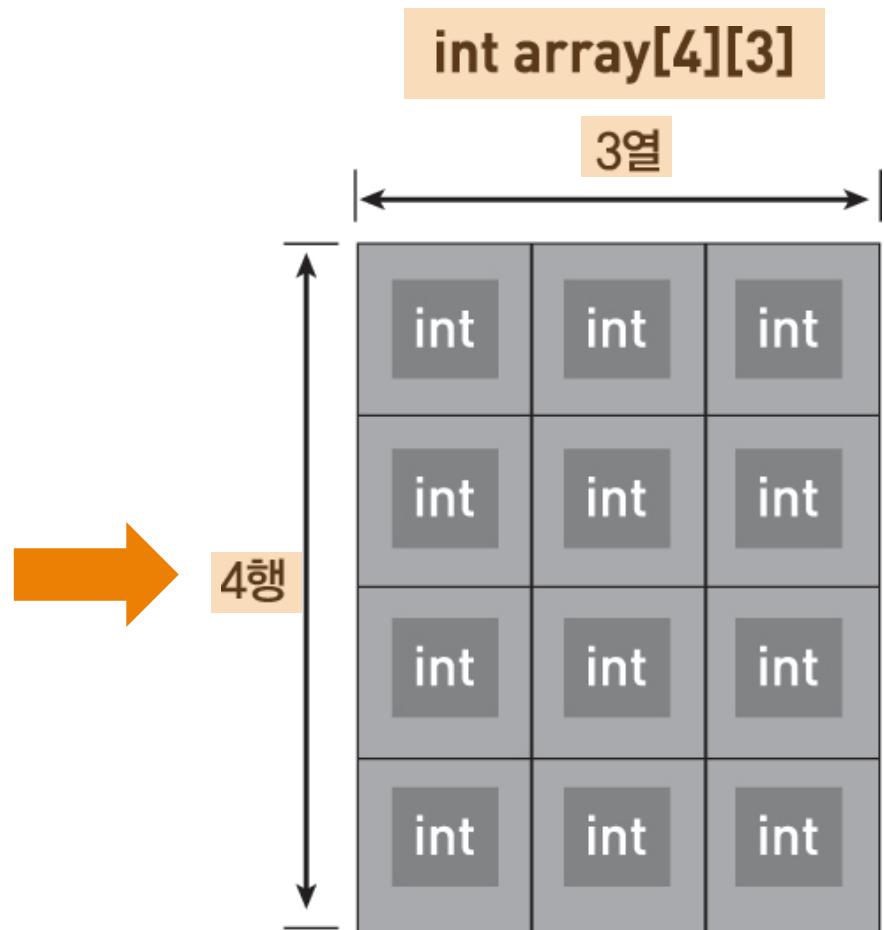
## 2.1 다차원 배열이란 (3/14)

### ▶ 2차원 배열의 선언

- ✓ 행과 열을 지정

```
#include <stdio.h>
int main(void)
{
    // 2차원 배열의 선언
    int array[4][3];

    return 0;
}
```



4(int) \* 4 \* 3는 48바이트

## 2.1 다차원 배열이란 (4/14)---[2-1.c 실습]

```
#include<stdio.h>
int main(void)
{
    // 2차원 배열의 선언
    int array[4][3]; // 4행 3열의 배열 길이 선언

    array[0][0]=1; array[0][1]=2; array[0][2]=3;
    array[1][0]=4; array[1][1]=5; array[1][2]=6;
    array[2][0]=7; array[2][1]=8; array[2][2]=9;
    array[3][0]=10; array[3][1]=11; array[3][2]=12;

    printf("%d %d %d \n",array[0][0], array[0][1], array[0][2]); // 0행 출력
    printf("%d %d %d \n",array[1][0], array[1][1], array[1][2]); // 1행 출력
    printf("%d %d %d \n",array[2][0], array[2][1], array[2][2]); // 2행 출력
    printf("%d %d %d \n",array[3][0], array[3][1], array[3][2]); // 3행 출력
    return 0;
}
```



## 2.1 다차원 배열이란 (5/14)---[2-1.c 분석]

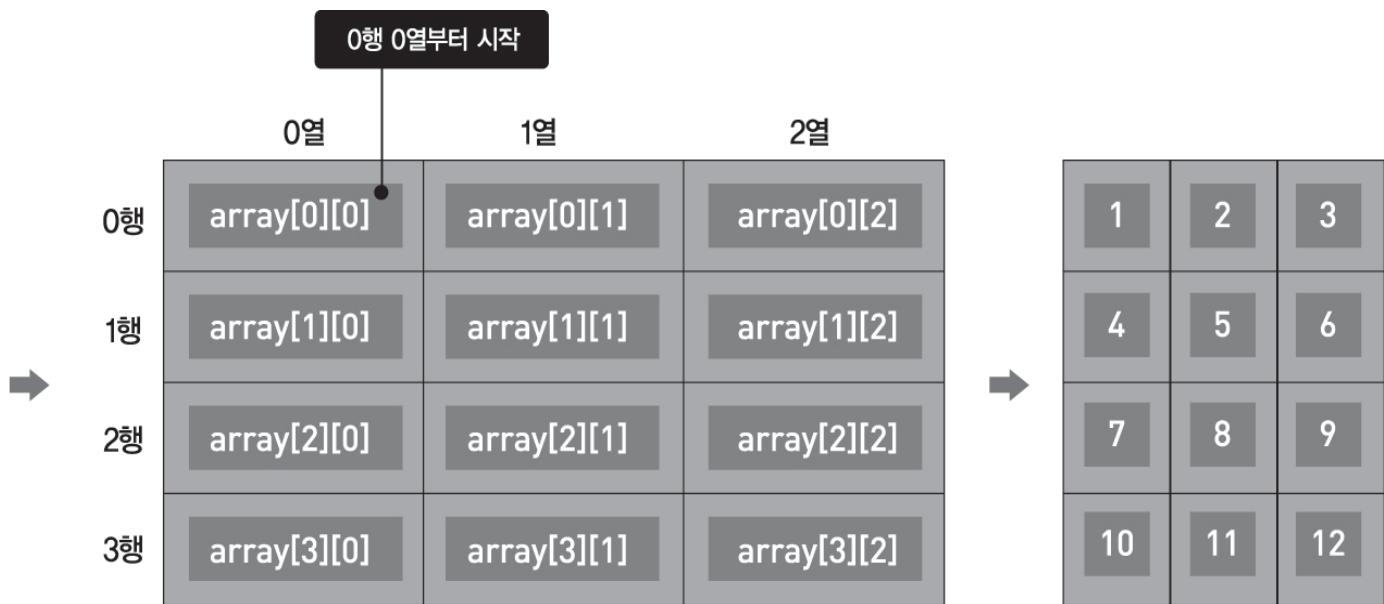
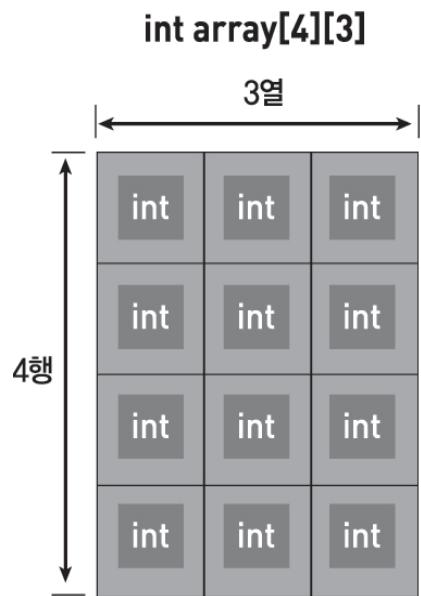
**int array[4][3];**

array[0][0]=1; array[0][1]=2; array[0][2]=3; // 0행의 배열 요소들에 데이터 저장

array[1][0]=4; array[1][1]=5; array[1][2]=6; // 1행의 배열 요소들에 데이터 저장

array[2][0]=7; array[2][1]=8; array[2][2]=9; // 2행의 배열 요소들에 데이터 저장

array[3][0]=10; array[3][1]=11; array[3][2]=12; // 3행의 배열 요소들에 데이터 저장



4(int) \* 4 \* 3는 48바이트

4(int) \* 4(행) \* 3(열)는 48바이트

## 2.1 다차원 배열이란 (6/14)---[2-2.c 실습]

```
#include<stdio.h>
int main(void)
{
    // 2차원 배열 선언 과 동시에 데이터 저장(초기화)
    int array1[4][3]={1,2,3,4,5,6,7,8,9,10,11,12};
    int array2[4][3]={1,2,3,4,5};

    // 2차원 배열 array1의 데이터 출력
    printf("%d %d %d \n",array1[0][0], array1[0][1], array1[0][2]); // 0행 출력
    printf("%d %d %d \n",array1[1][0], array1[1][1], array1[1][2]); // 1행 출력
    printf("%d %d %d \n",array1[2][0], array1[2][1], array1[2][2]); // 2행 출력
    printf("%d %d %d \n",array1[3][0], array1[3][1], array1[3][2]); // 3행 출력

    printf("-----\n");

    // 2차원 배열 array2의 데이터 출력
    printf("%d %d %d \n",array2[0][0], array2[0][1], array2[0][2]); // 0행 출력
    printf("%d %d %d \n",array2[1][0], array2[1][1], array2[1][2]); // 1행 출력
    printf("%d %d %d \n",array2[2][0], array2[2][1], array2[2][2]); // 2행 출력
    printf("%d %d %d \n",array2[3][0], array2[3][1], array2[3][2]); // 3행 출력

    return 0;
}
```

## 2.1 다차원 배열이란 (7/14)---[2-2.c 분석]

`int array1[4][3]`

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

`int array2[4][3]`

1	2	3
4	5	0
0	0	0
0	0	0

## 2.1 다차원 배열이란 (8/14)---[2-3.c 실습]

```
#include<stdio.h>
int main(void)
{
    int array1[4][3]={{1,2},{3},{4},{5}};
    int array2[4][3]={{1,2,3},{4,5,6},{7,8,9},{10}};

    printf("%d %d %d \n",array1[0][0], array1[0][1], array1[0][2]);
    printf("%d %d %d \n",array1[1][0], array1[1][1], array1[1][2]);
    printf("%d %d %d \n",array1[2][0], array1[2][1], array1[2][2]);
    printf("%d %d %d \n",array1[3][0], array1[3][1], array1[3][2]);

    printf("-----\n");

    printf("%d %d %d \n",array2[0][0], array2[0][1], array2[0][2]);
    printf("%d %d %d \n",array2[1][0], array2[1][1], array2[1][2]);
    printf("%d %d %d \n",array2[2][0], array2[2][1], array2[2][2]);
    printf("%d %d %d \n",array2[3][0], array2[3][1], array2[3][2]);
    return 0;
}
```

## 2.1 다차원 배열이란 (9/14)---[2-3.c 분석]

// 행 단위로 2차원 배열의 선언과 동시 초기화



행 단위 초기화

```
int array1[4][3]={{1, 2}, {3}, {4}, {5}}
```

1	2	0
3	0	0
4	0	0
5	0	0

행 단위 초기화

```
int array2[4][3]={{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {10}}
```

1	2	3
4	5	6
7	8	9
10	0	0

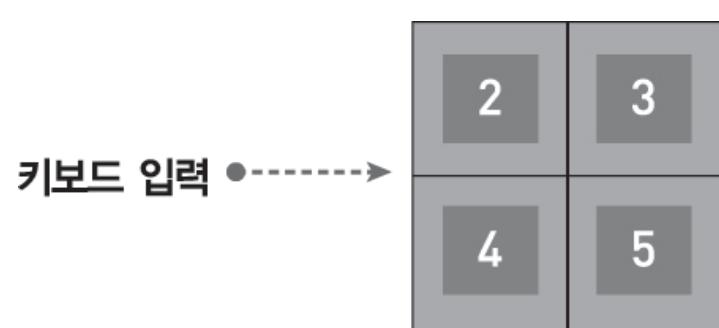
## 2.1 다차원 배열이란 (10/14)---[2-4.c 실습]

```
#include<stdio.h>
int main(void)
{
    // 2차원 배열의 선언
    int array[2][2];
    int i,j;

    // 2차원 배열에 데이터 입력
    for(i=0;i<2;i++)
    {
        for(j=0; j<2; j++)
        {
            printf("정수를 입력하세요: ");
            scanf("%d", &array[i][j]);
        }
    }
}
```

//2차원 배열에 데이터 출력

```
for(i=0;i<2;i++)
{
    for(j=0; j<2; j++)
    {
        printf("%3d",array[i][j]);
    }
    printf("\n");
}
return 0;
}
```



## 2.1 다차원 배열이란 (11/14)

### ▶ 2차원 배열 선언 시 주의사항

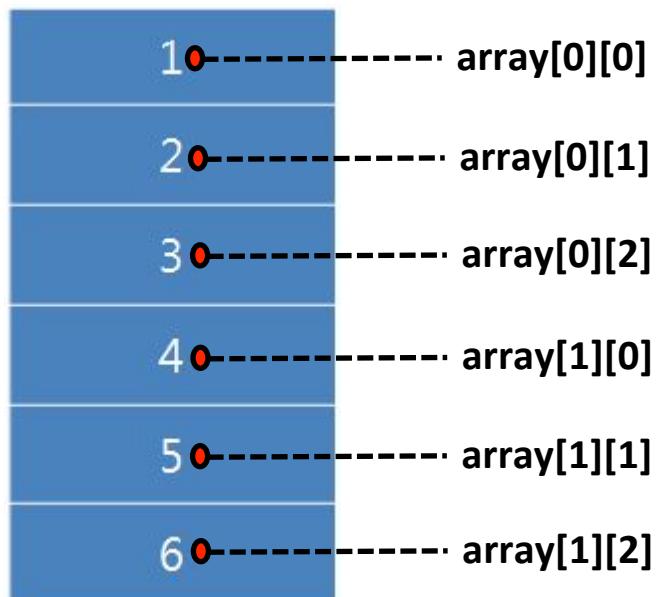
- ✓ 열의 길이는 반드시 설정

```
int array1[ ][ ]={1, 2, 3, 4, 5, 6, ,7 ,8 ,9, 10, 11, 12}; // 에러
int array2[4][ ]={1, 2, 3, 4, 5, 6, ,7 ,8 ,9, 10, 11, 12}; // 에러
int array3[ ][3]={1, 2, 3, 4, 5, 6, ,7 ,8 ,9, 10, 11, 12}; // 정상 int array
4[ ][4]={1, 2, 3, 4, 5, 6, ,7 ,8 ,9, 10, 11, 12}; // 정상
int array5[ ][2]={1, 2, 3, 4, 5, 6, ,7 ,8 ,9, 10, 11, 12}; // 정상
```

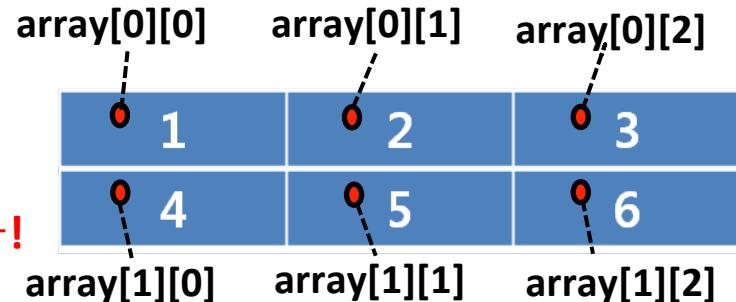
## 2.1 다차원 배열이란 (12/14)

### ▶ 2차원 배열의 물리적 메모리 구조

```
int array1[2][3] = {1, 2, 3, 4, 5, 6};
```



이렇게 이해하자!



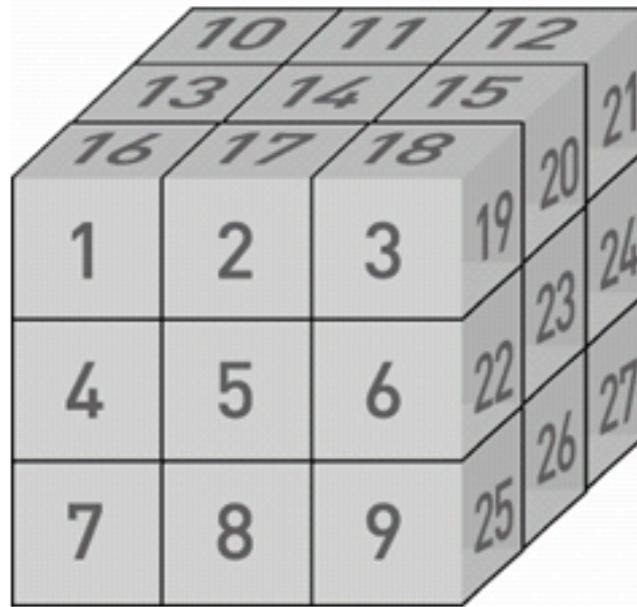
2차원 메모리 구조

물리적 메모리 구조

## 2.1 다차원 배열이란 (13/14)

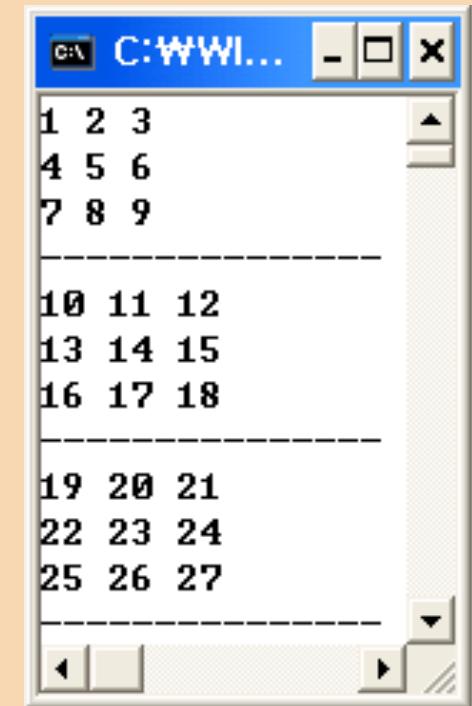
### ▶ 3차원 배열의 이해

면      행      열  
↓      ↓      ↓  
`int array[3][3][3]`



## 2.1 다차원 배열이란 (14/14)---[2-5.c 실습]

```
int i, j, k;
int array[3][3][3]={ {1,2,3,4,5,6,7,8,9},
                      {10,11,12,13,14,15,16,17,18},
                      {19,20,21,22,23,24,25,26,27}
                    };
for(i=0; i<3; i++) // 0면, 1면, 2면
{
    for(j=0; j<3; j++) // 0행, 1행, 2행
    {
        for(k=0; k<3; k++) // 0열, 1열, 2열
        {
            printf("%d ", array[i][j][k]);
        }
        printf("\n");
    }
    printf("-----\n");
}
```



## 2.2 2차원 배열의 주소와 값의 참조

## 2.2 2차원 배열의 주소와 값의 참조 (1/19)

### ▶ &연산자

- ✓ ‘2차원 배열 요소의 주소를 참조하는 연산자이다.’

2차원 배열의 주소 표현

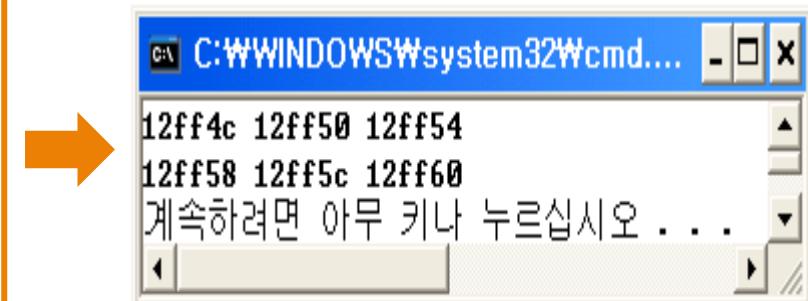
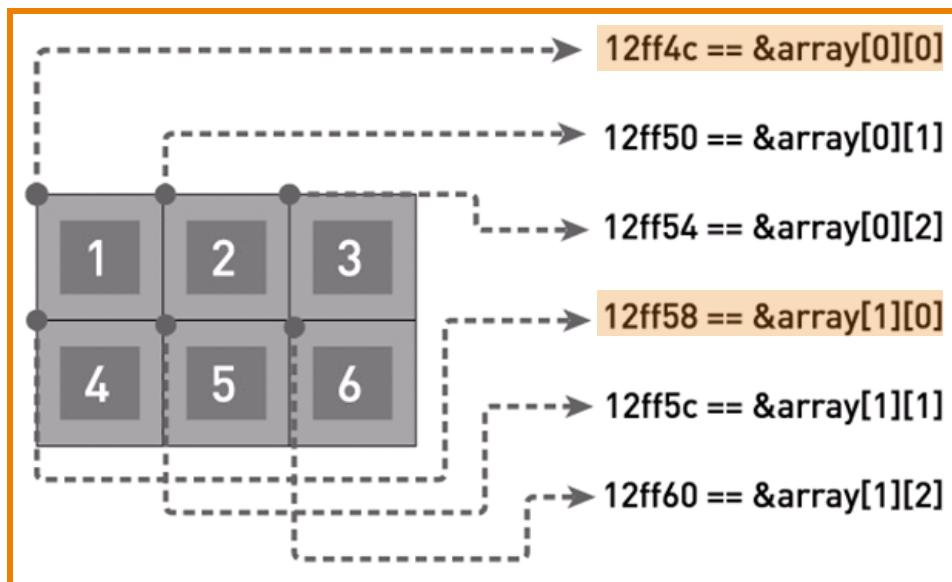
사용법: &2차원 배열 요소

```
int array[2][2]={10,20,30,40};
```

```
printf("%x %x \n", &array[0][0], &array[0][1]);
printf("%x %x \n", &array[1][0], &array[1][1]);
```

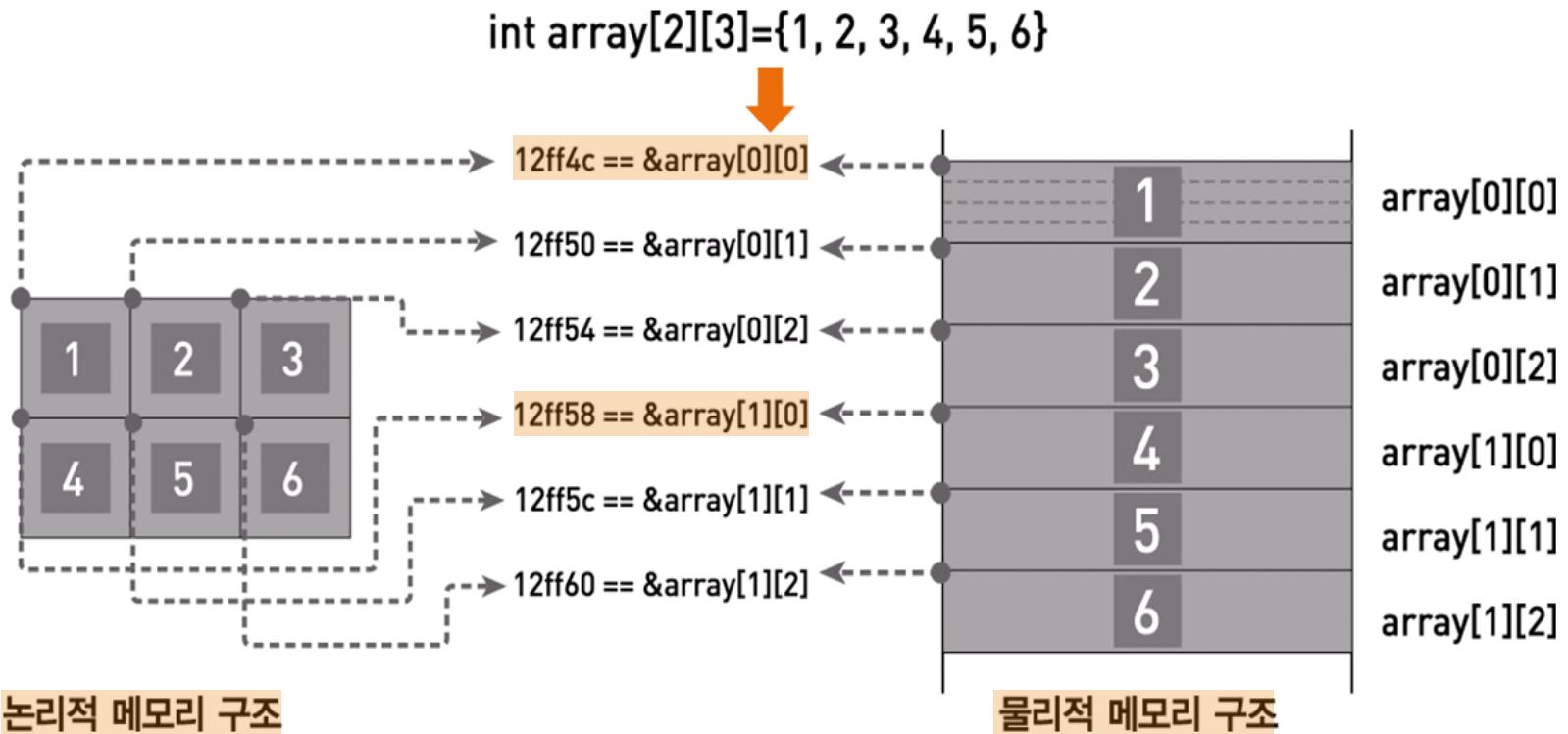
## 2.2 2차원 배열의 주소와 값의 참조 (2/19)---[2-6.c 실습]

```
#include<stdio.h>
int main(void)
{
    int array[2][3]={1,2,3,4,5,6};
    printf("%x %x %x \n", &array[0][0],&array[0][1],&array[0][2]);
    printf("%x %x %x \n", &array[1][0],&array[1][1],&array[1][2]);
    return 0;
}
```



## 2.2 2차원 배열의 주소와 값의 참조 (3/19)---[2-6.c 분석]

```
int array[2][3]={1,2,3,4,5,6};  
printf("%x %x %x \n", &array[0][0],&array[0][1],&array[0][2]);  
printf("%x %x %x \n", &array[1][0],&array[1][1],&array[1][2]);
```



## 2.2 2차원 배열의 주소와 값의 참조 (4/19)

### ▶ 2차원 배열의 다양한 주소 표현

- ① '2차원 배열 이름은 2차원 배열의 시작 주소이다.'
- ② '2차원 배열의 행의 요소는 행을 대표하는 주소이다.'
- ③ '2차원 배열에서  $\text{array}[i] == *(\text{array}+i)$ 는 주소이다.'

## 2.2 2차원 배열의 주소와 값의 참조 (5/19)---[2-7.c 실습]

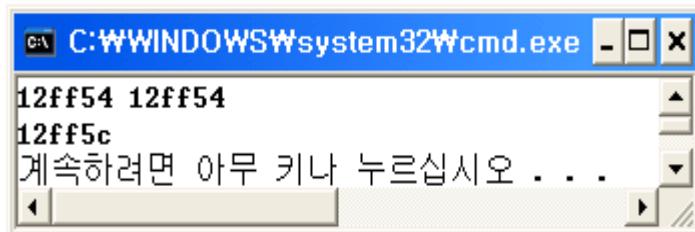
### ▶ 2차원 배열의 다양한 주소 표현

① '2차원 배열 이름은 2차원 배열의 시작 주소이다.'

```
#include<stdio.h>
int main(void)
{
    int array[2][2] = {10,20,30,40};

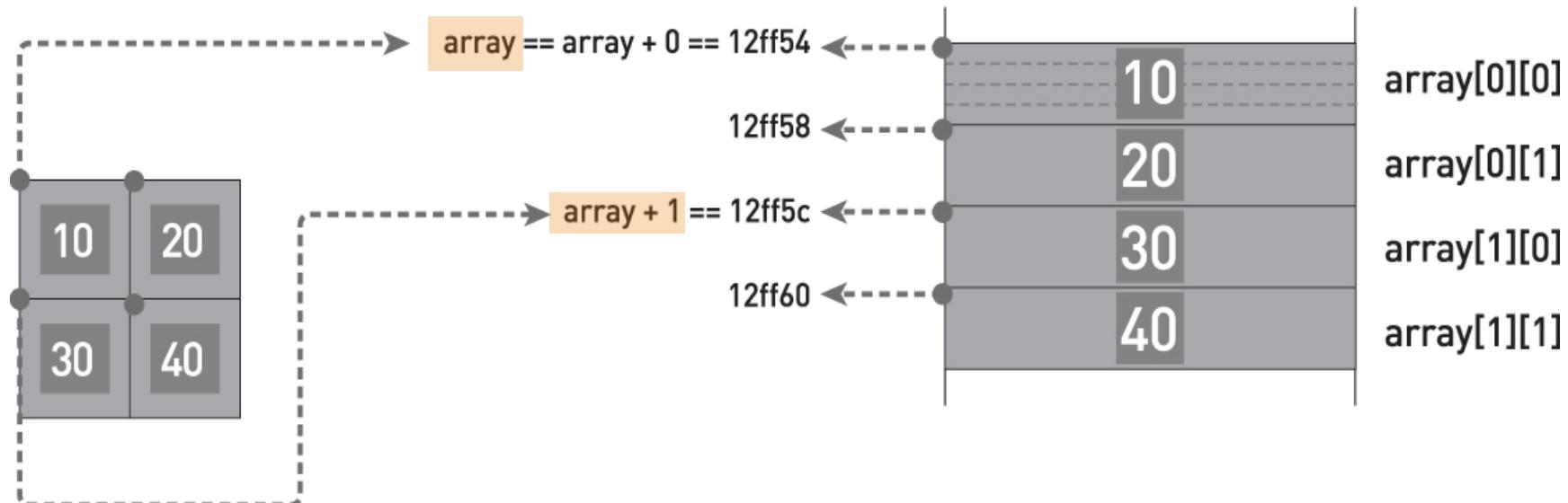
    printf("%x %x \n", array, array+0);          // 0행의 주소
    printf("%x \n", array+1);                      // 1행의 주소

    return 0;
}
```



## 2.2 2차원 배열의 주소와 값의 참조 (6/19)---[2-7.c 분석]

int array[2][2]={10, 20, 30, 40}



논리적 메모리 구조

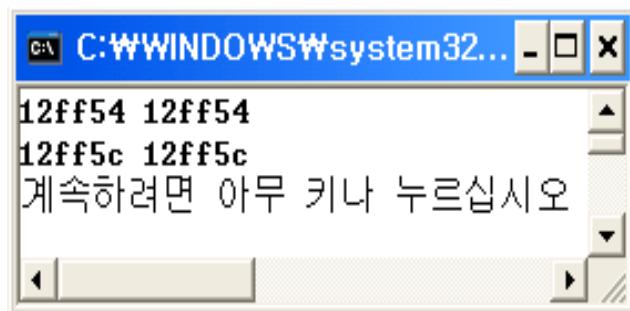
물리적 메모리 구조

## 2.2 2차원 배열의 주소와 값의 참조 (7/19)---[2-8.c 실습]

### ▶ 2차원 배열의 다양한 주소 표현

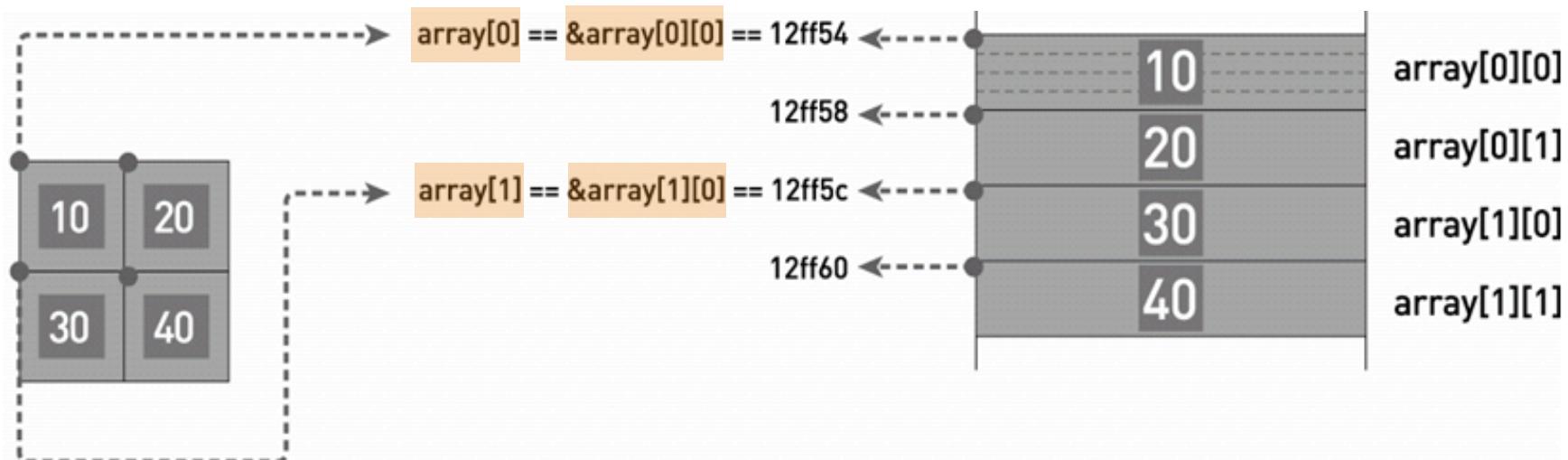
② '2차원 배열의 행의 요소는 행을 대표하는 주소이다.'

```
#include<stdio.h>
int main(void)
{
    int array[2][2] = {10,20,30,40};
    printf("%x %x\n", array[0], &array[0][0]);
    printf("%x %x\n", array[1], &array[1][0]);
    return 0;
}
```



## 2.2 2차원 배열의 주소와 값의 참조 (8/19)---[2-8.c 분석]

`int array[2][2] = {10, 20, 30, 40};`



논리적 메모리 구조

물리적 메모리 구조

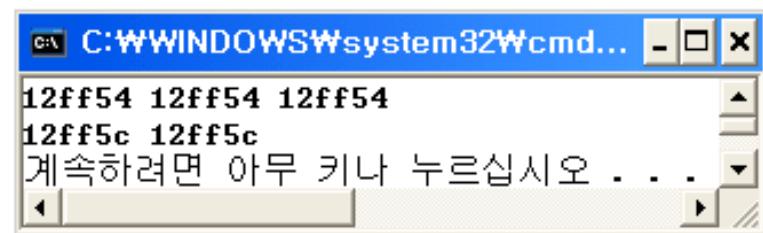
## 2.2 2차원 배열의 주소와 값의 참조 (9/19)---[2-9.c 실습]

### ▶ 2차원 배열의 다양한 주소 표현

③ '2차원 배열에서 **array[i] == \*(array+i)**는 주소이다.'

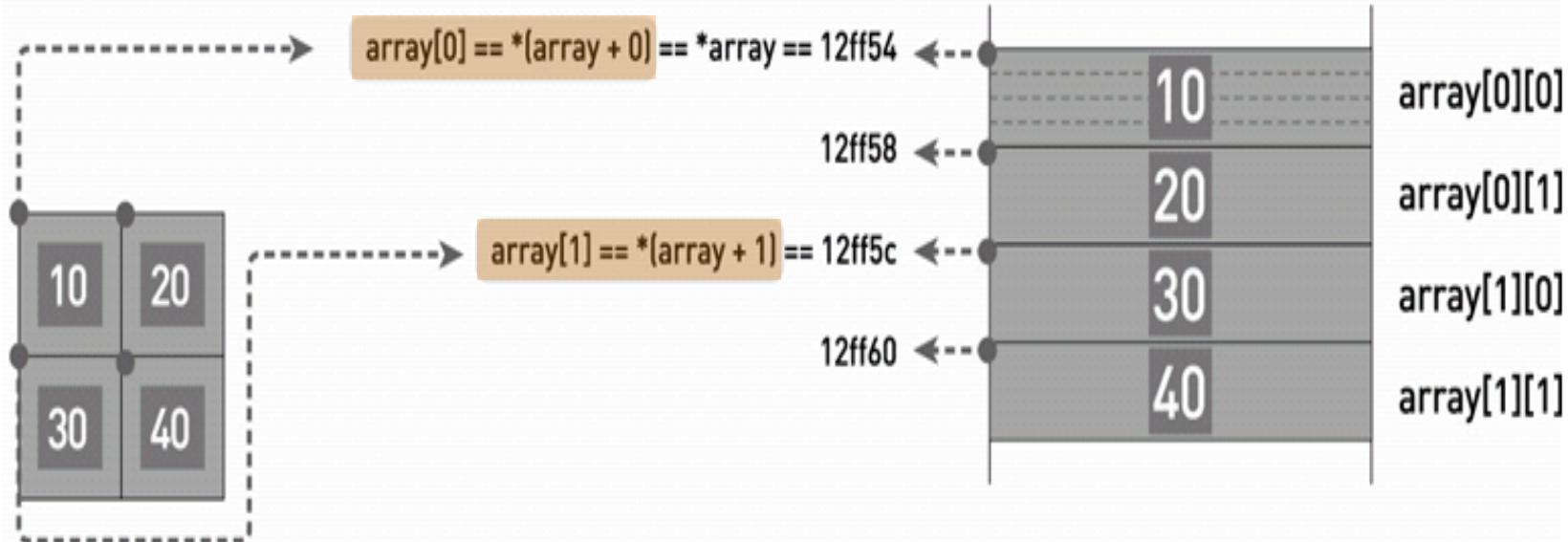
```
#include<stdio.h>
int main(void)
{
    int array[2][2] = {10,20,30,40};

    printf("%x %x %x\n", array[0],*(array+0), *array);
    printf("%x %x \n", array[1],*(array+1));
    return 0;
}
```



## 2.2 2차원 배열의 주소와 값의 참조 (10/19)---[2-9.c 분석]

`int array[2][2] = {10, 20, 30, 40};`



논리적 메모리 구조

물리적 메모리 구조

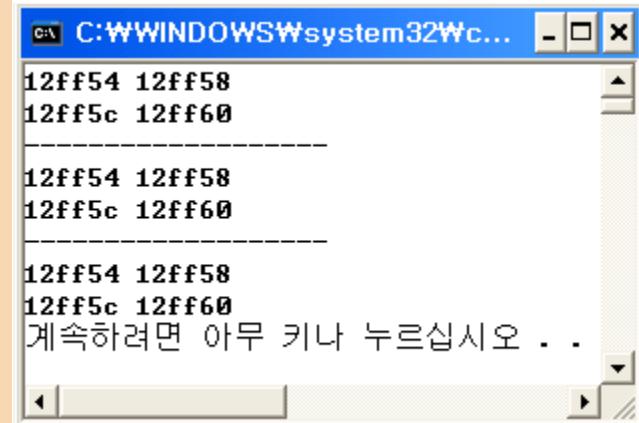
## 2.2 2차원 배열의 주소와 값의 참조 (11/19)---[2-10.c 실습]

```
#include<stdio.h>
int main(void)
{
    int array[2][2] = {10,20,30,40};

    printf("%x %x \n", &array[0][0],&array[0][1]);
    printf("%x %x \n", &array[1][0],&array[1][1]);

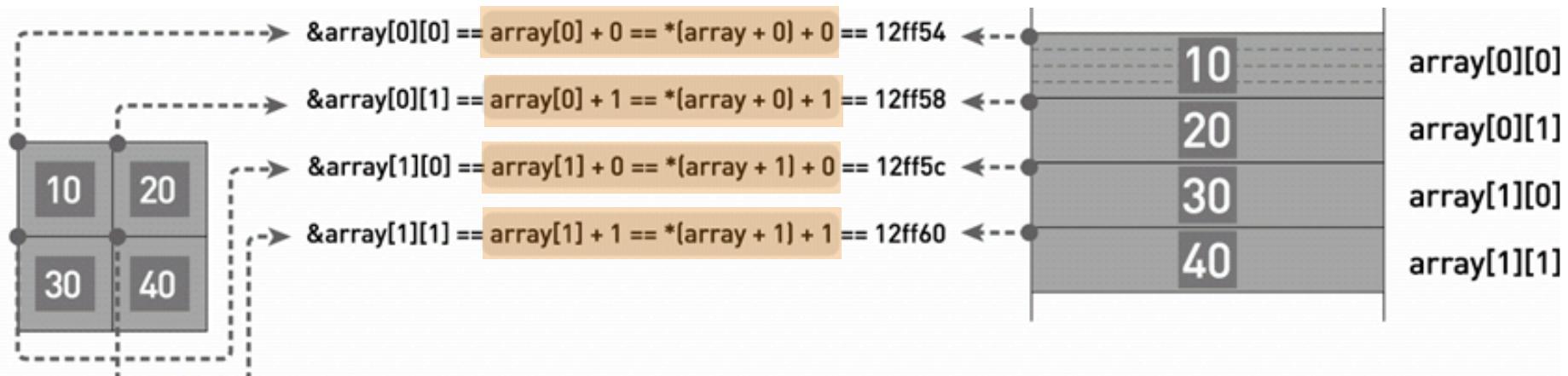
    printf("-----\n");
    printf("%x %x \n", array[0]+0, array[0]+1);
    printf("%x %x \n", array[1]+0, array[1]+1);

    printf("-----\n");
    printf("%x %x \n", *(array+0)+0, *(array+0)+1);
    printf("%x %x \n", *(array+1)+0, *(array+1)+1);
    return 0;
}
```



## 2.2 2차원 배열의 주소와 값의 참조 (12/19)---[2-10.c 분석]

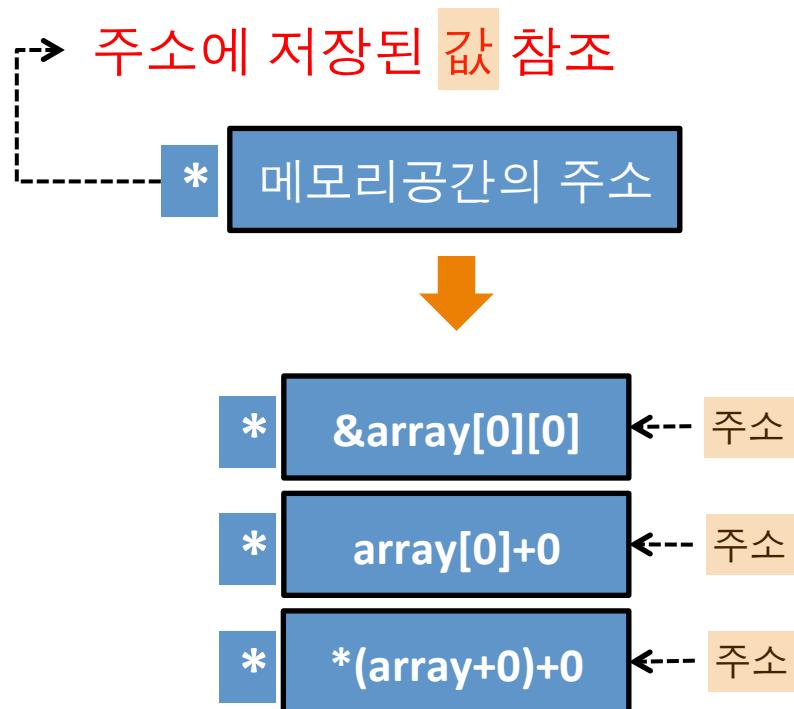
```
int array[2][2] = {10,20,30,40};
```



## 2.2 2차원 배열의 주소와 값의 참조 (13/19)

### \*연산자

✓ '2차원 배열 요소에 저장된 값을 참조하는 연산자 이다.'



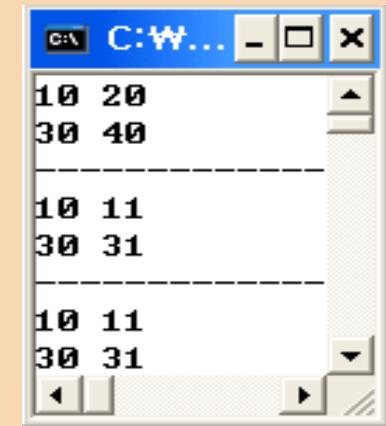
## 2.2 2차원 배열의 주소와 값의 참조 (14/19)---[2-11.c 실습]

```
#include<stdio.h>
int main(void)
{
    int array[2][2] = {10,20,30,40};

    printf("%d %d \n", *&array[0][0],*&array[0][1]);
    printf("%d %d \n", *&array[1][0],*&array[1][1]);

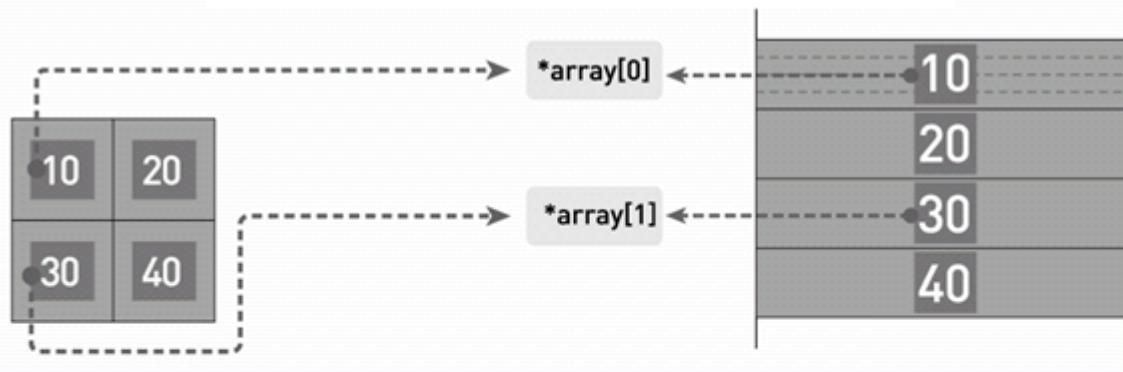
    printf("-----\n");
    printf("%d %d \n", *array[0]+0, *array[0]+1 );
    printf("%d %d \n", *array[1]+0, *array[1]+1 );

    printf("-----\n");
    printf("%d %d \n", **(array+0)+0, **(array+0)+1 );
    printf("%d %d \n", **(array+1)+0, **(array+1)+1 );
    return 0;
}
```



## 2.2 2차원 배열의 주소와 값의 참조 (15/19)---[2-11.c 분석]

```
printf("%d %d \n", *array[0]+0, *array[0]+1 );
printf("%d %d \n", *array[1]+0, *array[1]+1 );
```



$$\text{array}[i] \\ == \\ *(\text{array}+i)$$

$$\frac{*array[0] + 0}{10}$$

↑  
① ②

$$\frac{*array[0] + 1}{10}$$

↑  
① ②

$$\frac{*array[1] + 0}{30}$$

↑  
① ②

$$\frac{*array[1] + 1}{30}$$

↑  
① ②

출력결과: 10

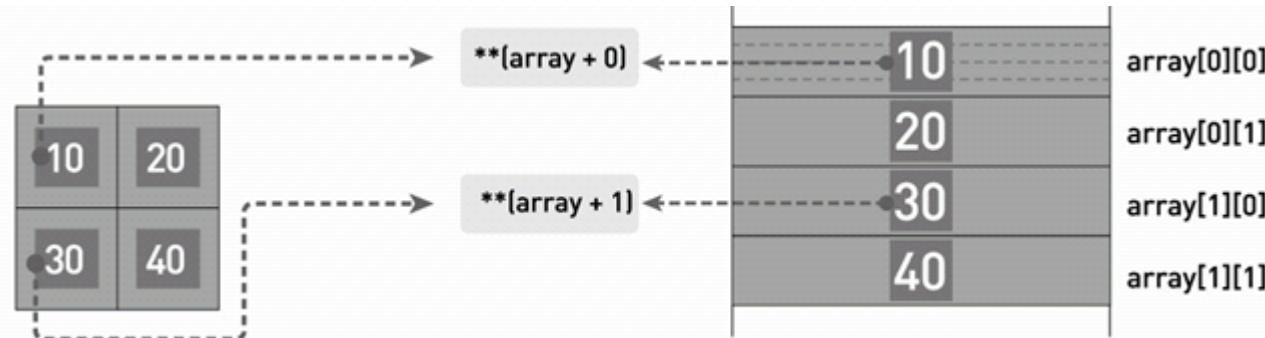
출력결과: 11

출력결과: 30

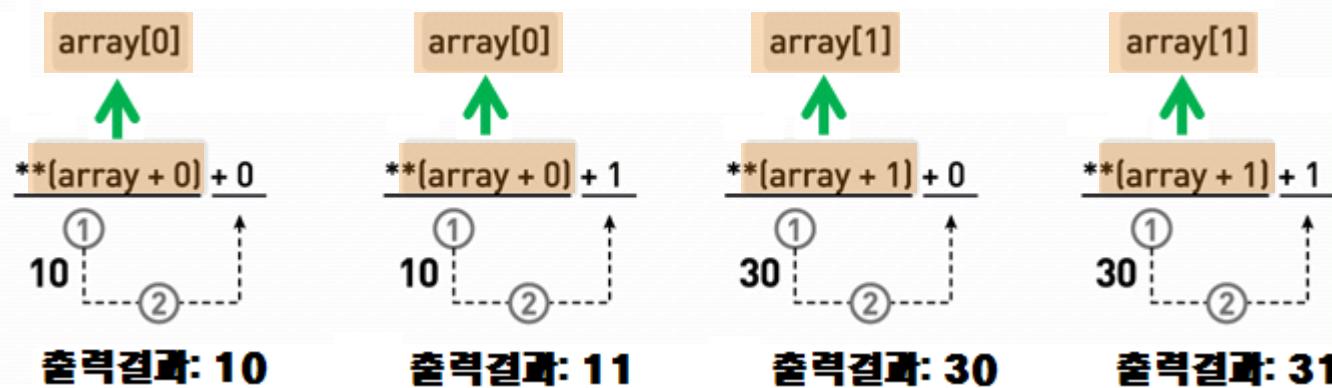
출력결과: 31

## 2.2 2차원 배열의 주소와 값의 참조 (16/19)---[2-11.c 분석]

```
printf("%d %d \n", **(array+0)+0, **(array+0)+1 );  
printf("%d %d \n", **(array+1)+0, **(array+1)+1 );
```



array[i]  
==  
\*(array+i)



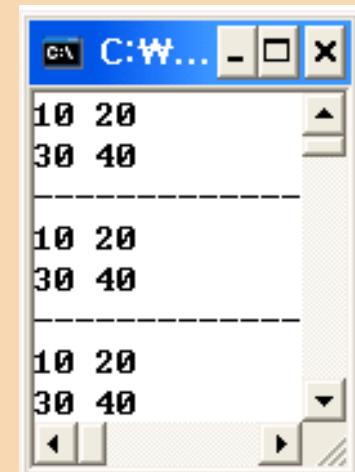
## 2.2 2차원 배열의 주소와 값의 참조 (17/19)---[2-12.c 실습]

```
#include<stdio.h>
int main(void)
{
    int array[2][2] = {10,20,30,40};

    printf("%d %d \n", *array[0][0],*array[0][1]);
    printf("%d %d \n", *array[1][0],*array[1][1]);

    printf("-----\n");
    printf("%d %d \n", *(array[0]+0), *(array[0]+1));
    printf("%d %d \n", *(array[1]+0), *(array[1]+1));

    printf("-----\n");
    printf("%d %d \n", *(*array+0)+0, *(*array+0)+1));
    printf("%d %d \n", *(*array+1)+0, *(*array+1)+1));
    return 0;
}
```

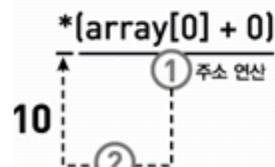
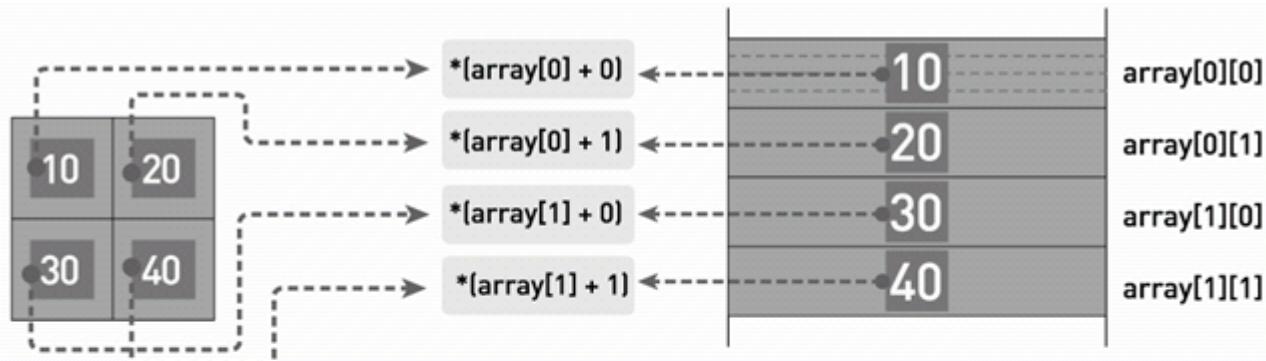


## 2.2 2차원 배열의 주소와 값의 참조 (18/19)---[2-12.c 분석]

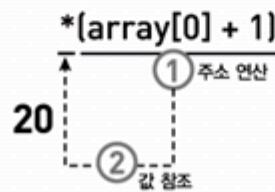
```
printf("%d %d \n", *(array[0]+0), *(array[0]+1));  
printf("%d %d \n", *(array[1]+0), *(array[1]+1));
```



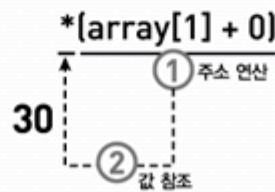
$$\text{array}[i] \\ == \\ *(\text{array}+i)$$



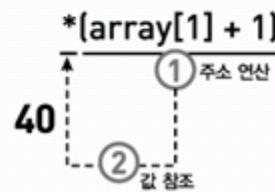
출력결과: 10



출력결과: 20



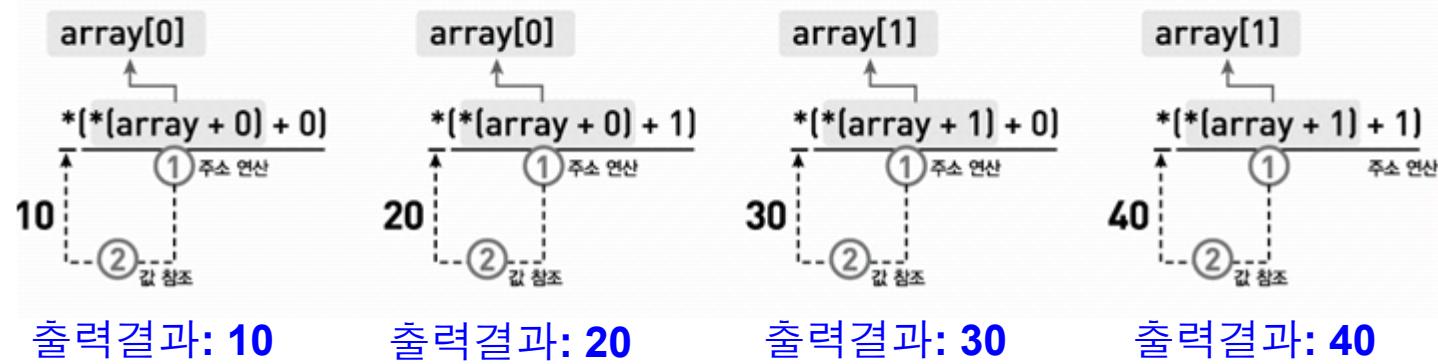
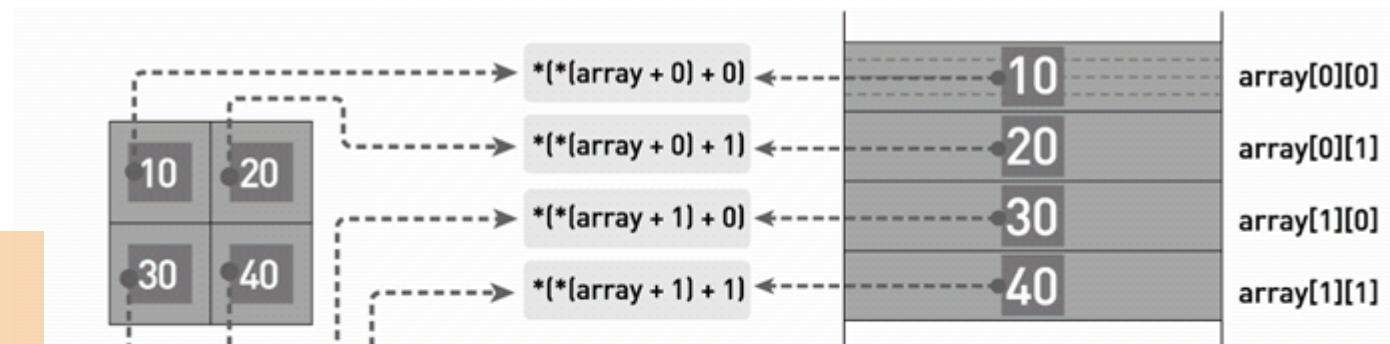
출력결과: 30



출력결과: 40

## 2.2 2차원 배열의 주소와 값의 참조 (19/19)---[2-12.c 분석]

```
printf("%d %d \n", *(*array+0)+0, *(*array+0)+1);
printf("%d %d \n", *(*array+1)+0, *(*array+1)+1);
```



## 공부한 내용 떠올리기

- ▶ 2차원 배열의 선언과 구성 요소
- ▶ 2차원 배열에 데이터를 저장하는 방법
- ▶ 2차원 배열을 선언할 때 주의할 사항
- ▶ 2차원 배열의 주소와 값을 참조하는 다양한 방법