

# 윤성우의 열혈 C 프로그래밍



윤성우 저 열혈강의 C 프로그래밍 개정판

Chapter 16. 다차원 배열

# 윤성우의 열혈 C 프로그래밍



## Chapter 16-1. 다차원 배열의 이해와 활용

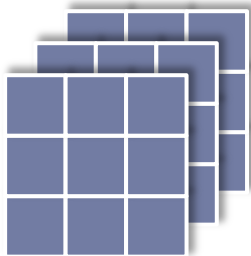
윤성우 저 열혈강의 C 프로그래밍 개정판

## 2차원, 3차원 배열 OK! 4차원, 5차원 배열 NO!

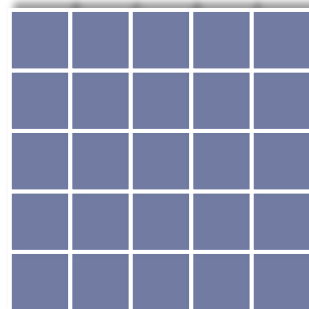
<code>int arrOneDim[10];</code>	길이가 10인 1차원 int형 배열
<code>int arrTwoDim[5][5];</code>	가로, 세로의 길이가 각각 5인 2차원 int형 배열
<code>int arrThreeDim[3][3][3];</code>	가로, 세로, 높이의 길이가 각각 3인 3차원 int형 배열



1차원 배열 *arrOneDim*



3차원 배열 *arrThreeDim*



2차원 배열 *arrTwoDim*

문법적으로는 4차원 5차원 배열의 선언도 가능  
하지만 그것은 의미를 부여하기 힘든, 의미가  
없는 배열이다.

# 다차원 배열을 의미하는 2차원 배열의 선언

2차원 배열의 선언 방식 → **TYPE arr[세로길이][가로길이];**

	1열	2열	3열	4열
1행	[0][0]	[0][1]	[0][2]	[0][3]
2행	[1][0]	[1][1]	[1][2]	[1][3]
3행	[2][0]	[2][1]	[2][2]	[2][3]

*int arr1[3][4];*

	1열	2열	3열	4열	5열	6열
1행	[0][0]	[0][1]	[0][2]	[0][3]	[0][4]	[0][5]
2행	[1][0]	[1][1]	[1][2]	[1][3]	[1][4]	[1][5]

*int arr2[2][6];*

```
int main(void)
{
    int arr1[3][4];
    int arr2[7][9];
    printf("세로3, 가로4: %d \n", sizeof(arr1));
    printf("세로7, 가로9: %d \n", sizeof(arr2));
    return 0;
}
```

실행결과

세로3, 가로4: 48  
세로7, 가로9: 252

## 2차원 배열요소의 접근

```
int arr[3][3];
```



배열 생성

	1열	2열	3열
1행	0	0	0
2행	0	0	0
3행	0	0	0

```
arr[0][0]=1;
```



0 0 접근

	1열	2열	3열
1행	1	0	0
2행	0	0	0
3행	0	0	0

```
arr[0][1]=2;
```



0 1 접근

	1열	2열	3열
1행	1	2	0
2행	0	0	0
3행	0	0	0

일반화

```
arr[N-1][M-1]=20;  
printf("%d", arr[N-1][M-1]);
```

세로  $N$ , 가로  $M$ 의 위치에 값을 저장 및 참조

```
arr[2][1]=5;
```



2 1 접근

	1열	2열	3열
1행	1	2	0
2행	0	0	0
3행	0	5	0

## 2차원 배열요소 접근관련 예제

```
int main(void)
{
    int villa[4][2];
    int popu, i, j;
    /* 가구별 거주인원 입력 받기 */
    for(i=0; i<4; i++)
    {
        for(j=0; j<2; j++)
        {
            printf("%d층 %d호 인구수: ", i+1, j+1);
            scanf("%d", &villa[i][j]);
        }
    }

    /* 빌라의 층별 인구수 출력하기 */
    for(i=0; i<4; i++)
    {
        popu=0;
        popu += villa[i][0];
        popu += villa[i][1];
        printf("%d층 인구수: %d \n", i+1, popu);
    }
    return 0;
}
```

1층 1호 인구수: 2  
 1층 2호 인구수: 4  
 2층 1호 인구수: 3  
 2층 2호 인구수: 5  
 3층 1호 인구수: 2  
 3층 2호 인구수: 6  
 4층 1호 인구수: 4  
 4층 2호 인구수: 3  
 1층 인구수: 6  
 2층 인구수: 8  
 3층 인구수: 8  
 4층 인구수: 7

실행결과

## 2차원 배열의 메모리상 할당의 형태

0x1001번지, 0x1002번지, 0x1003번지, 0x1004번지, 0x1005번지 . . . .

1차원적 메모리의 주소 값

0x12-0x24번지, 0x12-0x25번지, 0x12-0x26번지, 0x12-0x27번지 . . . .

0x13-0x24번지, 0x13-0x25번지, 0x13-0x26번지, 0x13-0x27번지 . . . .

0x14-0x24번지, 0x14-0x25번지, 0x14-0x26번지, 0x14-0x27번지 . . . .

. . . .

2차원적 메모리의 주소 값

실제 메모리는 1차원의 형태로 주소 값이 지정이 된다.

따라서 아래와 같은 형태로 2차원 배열의 주소 값이 지정된다.

0x1000	0	arr[0][0]
0x1004	1	arr[0][1]
0x1008	2	arr[1][0]
0x100C	3	arr[1][1]
0x1010	4	arr[2][0]
0x1014	5	arr[2][1]

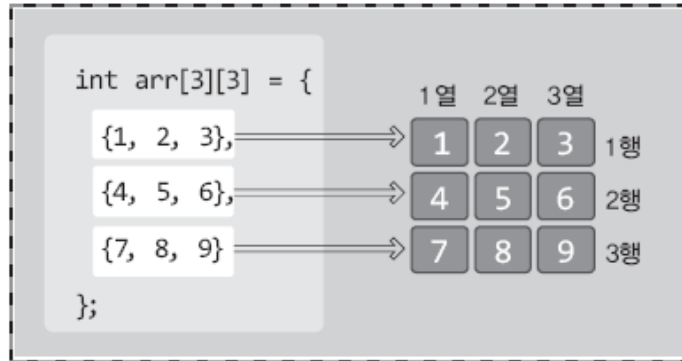
2차원 배열의  
실제 메모리  
할당형태

실행결과

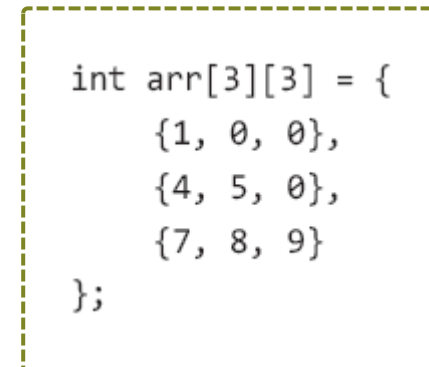
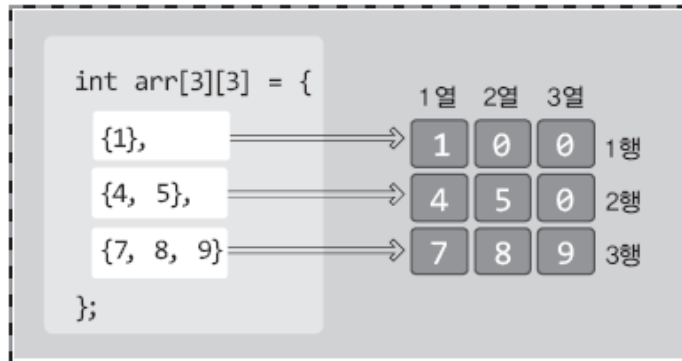
002AFD54  
002AFD58  
002AFD5C  
002AFD60  
002AFD64  
002AFD68

```
int main(void)
{
    int arr[3][2];
    int i, j;
    for(i=0; i<3; i++)
        for(j=0; j<2; j++)
            printf("%p \n", &arr[i][j]);
    return 0;
}
```

## 2차원 배열 선언과 동시에 초기화 하기



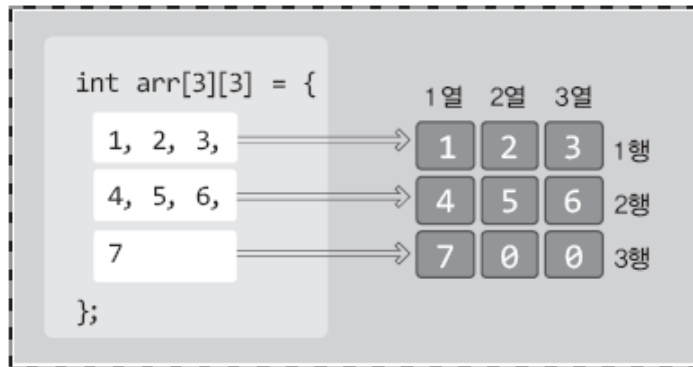
초기화 리스트 안에는 행 단위로 초기화할 값들을 별도의 중괄호로 명시한다.



채워지지 않은 빈 공간은 0으로 채워진다.



## 2차원 배열 선언과 동시에 초기화 하기2



별도의 중괄호를 사용하지 않으면 좌 상단부터 시작해서 우 하단으로 순서대로 초기화된다.



한 줄에 표현해도 된다.

```
int arr[3][3]={1, 2, 3, 4, 5, 6, 7};
```



마찬가지로 빈 공간은 0으로 채워진다.

```
int arr[3][3]={1, 2, 3, 4, 5, 6, 7, 0, 0};
```

## 2차원 배열 선언과 동시에 초기화 하기(예제)

```
int main(void)
{
    int i, j;

    /* 2차원 배열 초기화의 예 1 */
    int arr1[3][3]={
        {1, 2, 3},
        {4, 5, 6},
        {7, 8, 9}
    };

    /* 2차원 배열 초기화의 예 2 */
    int arr2[3][3]={
        {1},
        {4, 5},
        {7, 8, 9}
    };

    /* 2차원 배열 초기화의 예 3 */
    int arr3[3][3]={1, 2, 3, 4, 5, 6, 7};
```

```
    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
            printf("%d ", arr1[i][j]);
        printf("\n");
    }
    printf("\n");

    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
            printf("%d ", arr2[i][j]);
        printf("\n");
    }
    printf("\n");

    for(i=0; i<3; i++)
    {
        for(j=0; j<3; j++)
            printf("%d ", arr3[i][j]);
        printf("\n");
    }
    return 0;
}
```

실행결과

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

1 0 0
4 5 0
7 8 9

1 2 3
4 5 6
7 0 0
```



# 배열의 크기를 알려주지 않고 초기화하기

```
int arr[][]={1, 2, 3, 4, 5, 6, 7, 8};
```

8 by 1 ??

4 by 2 ??

2 by 4 ??

두 개가 모두 비면 컴파일러가 채워 넣을 숫자를 결정하지 못한다.

```
int arr1[][4]={1, 2, 3, 4, 5, 6, 7, 8};
```

```
int arr2[][2]={1, 2, 3, 4, 5, 6, 7, 8};
```

세로 길이만 생략할 수 있도록 약속되어 있다.



컴파일러가 세로 길이를 계산해 준다.

```
int arr1[2][4]={1, 2, 3, 4, 5, 6, 7, 8};
```

```
int arr2[4][2]={1, 2, 3, 4, 5, 6, 7, 8};
```

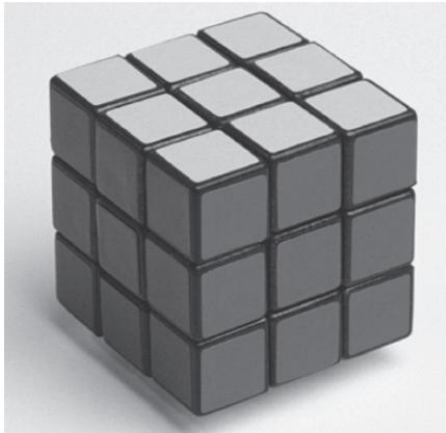
# 윤성우의 열혈 C 프로그래밍



## Chapter 16-2. 3차원 배열

윤성우 저 열혈강의 C 프로그래밍 개정판

## 3차원 배열의 논리적 구조



```
int main(void)
{
    int arr1[2][3][4];
    double arr2[5][5][5];
    printf("높이2, 세로3, 가로4 int형 배열: %d \n", sizeof(arr1));
    printf("높이5, 세로5, 가로5 double형 배열: %d \n", sizeof(arr2));
    return 0;
}
```

```
높이2, 세로3, 가로4 int형 배열: 96
높이5, 세로5, 가로5 double형 배열: 1000
```

실행결과

```
int arr1[2][3][4];
```

높이 2, 세로 3, 가로 4인 int형 3차원 배열(세로 3, 가로 4인 배열이 두 개 겹친 형태)

```
double arr2[5][5][5];
```

높이, 세로, 가로가 모두 5인 double형 3차원 배열(세로 5, 가로 5인 배열이 5개 겹친 형태)

## 3차원 배열의 선언과 접근

```
int main(void)
{
    int mean=0, i, j;
    int record[3][3][2]={
        {
            {70, 80}, // A 학급 학생 1의 성적
            {94, 90}, // A 학급 학생 2의 성적
            {70, 85}  // A 학급 학생 3의 성적
        },
        {
            {83, 90}, // B 학급 학생 1의 성적
            {95, 60}, // B 학급 학생 2의 성적
            {90, 82}  // B 학급 학생 3의 성적
        },
        {
            {98, 89}, // C 학급 학생 1의 성적
            {99, 94}, // C 학급 학생 2의 성적
            {91, 87}  // C 학급 학생 3의 성적
        }
    };
};
```

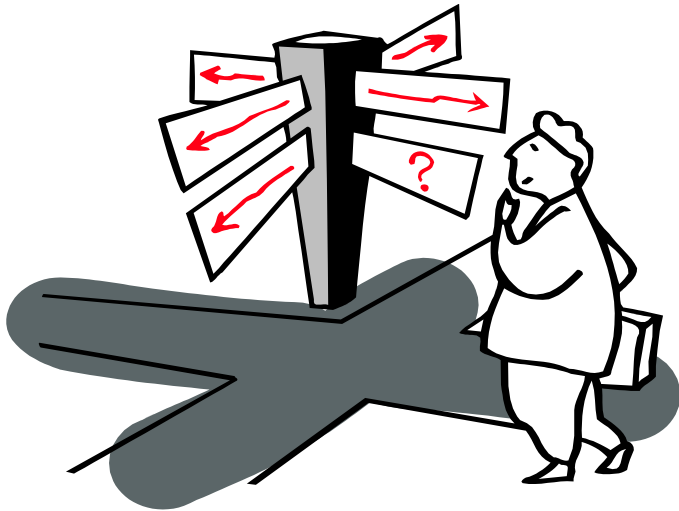
```
for(i=0; i<3; i++)
    for(j=0; j<2; j++)
        mean += record[0][i][j];
printf("A 학급 전체 평균: %g \n", (double)mean/6);

mean=0;
for(i=0; i<3; i++)
    for(j=0; j<2; j++)
        mean += record[1][i][j];
printf("B 학급 전체 평균: %g \n", (double)mean/6);

mean=0;
for(i=0; i<3; i++)
    for(j=0; j<2; j++)
        mean += record[2][i][j];
printf("C 학급 전체 평균: %g \n", (double)mean/6);
return 0;
}
```

A 학급 전체 평균: 81.5  
B 학급 전체 평균: 83.3333  
C 학급 전체 평균: 93

실행결과



Chapter 1b이 끝났습니다. 질문 있으신지요?