C 프로그래밍 프로젝트

Chap 16. 다차원 배열

2013.10.21.

오 병 우

컴퓨터공학과

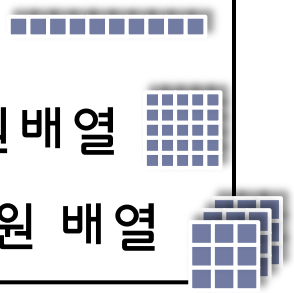
kit 국립금오공과대학교
Kumoh National Institute of Technology

16-1 2차원 배열

- 다차원 배열이란 무엇인가?
  - 2차원 이상의 배열을 의미함

- 다차원 배열의 선언

배열 선언 예	몇 차원 배열인가?
int arr[100]	1차원 배열
int arr[10][10]	10 × 10, 2차원배열
int arr[5][5][5]	5 × 5 × 5, 3차원 배열

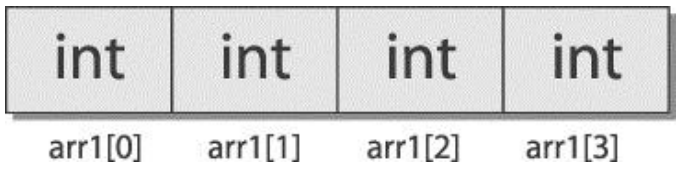


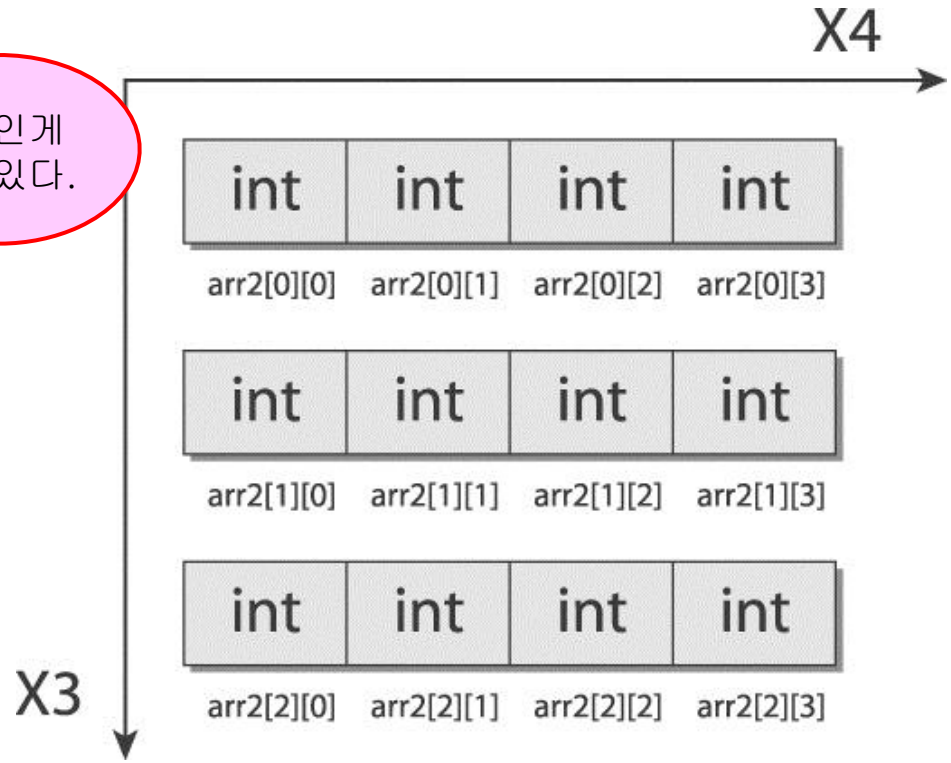
16-1 2차원 배열

- 2차원 배열의 선언
  - 2차원적 메모리 구조를 구성

```
int main(void)
{
   int arr1[4];
   int arr2[3][4];
   . . . . .
```

4개인게 3개 있다.





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

2차원 배열 요소의 접근 방법

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

배열 생성


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

00 접근


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

01 접근

일반화

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

세로 N, 가로 M의 위치에 값을 저장 및 참조

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

21 접근

16-1 2차원 배열

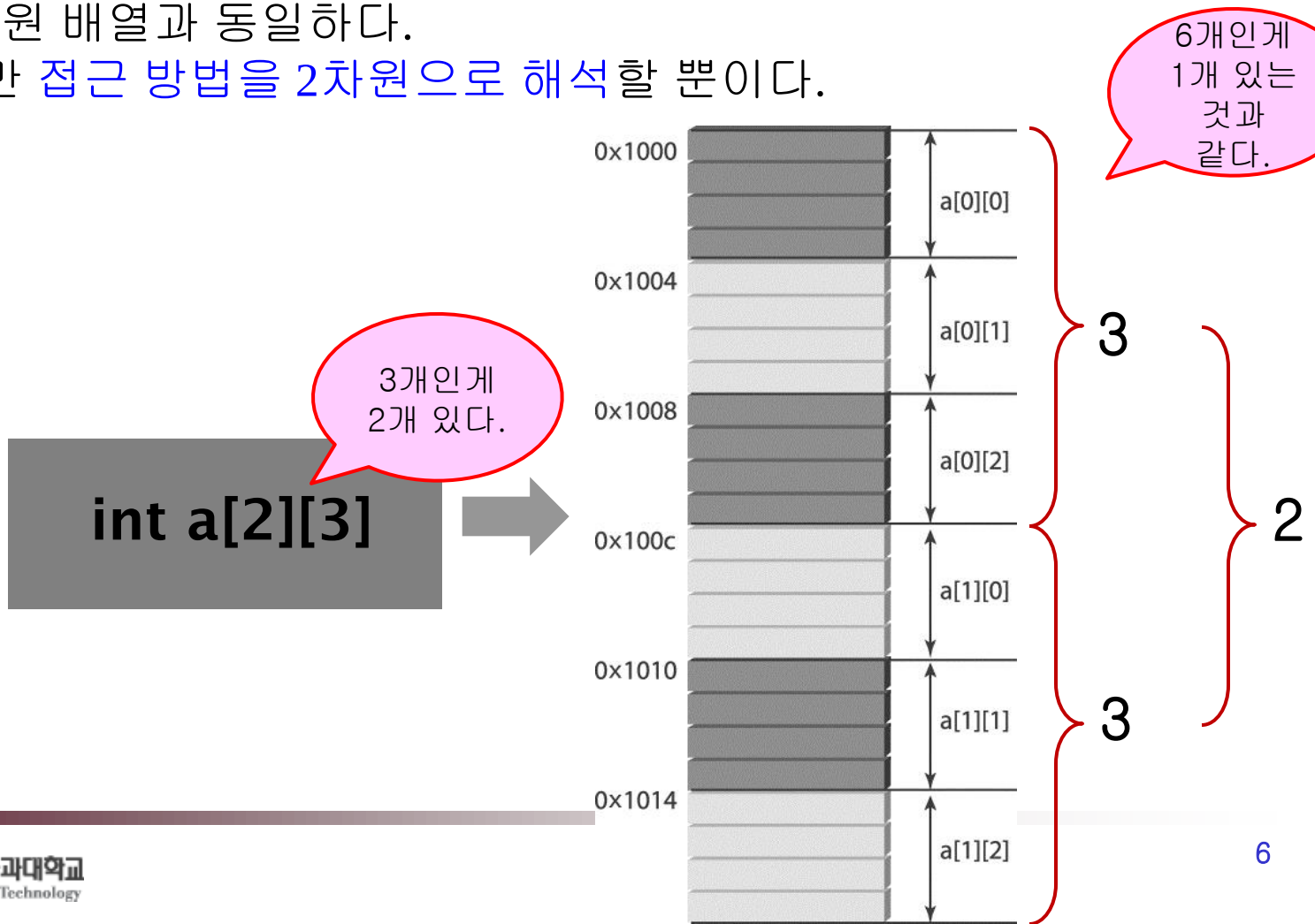
```
int main(void)
{
    int villa[4][2];
    int popu, i, j;

    /* 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

16-1 2차원 배열

- 다차원 배열의 실제 메모리 구성
  - 1차원 배열과 동일하다.
    다만 접근 방법을 2차원으로 해석할 뿐이다.



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.

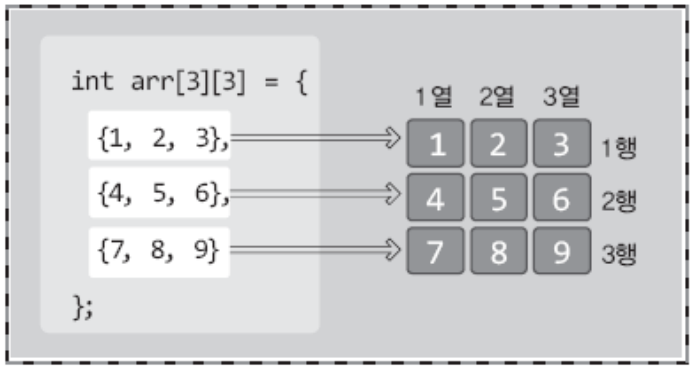
2차원 배열의 실제 메모리 할당형태

실행결과

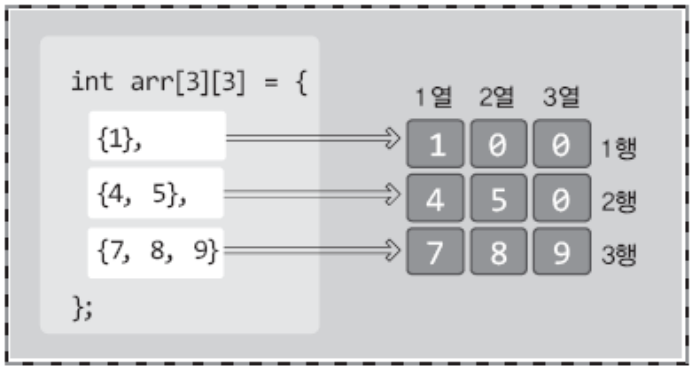
```
002AFD54
002AFD58
002AFD5C
002AFD60
002AFD64
002AFD68
```

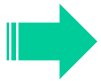
```
int main(void)
{
    int arr[3][2];
    int i, j;
    for(i=0; i<3; i++)
        for(j=0; j<2; j++)
            printf("%p \n", &arr[i][j]);
    return 0;
}
```

2차원 배열 초기화



초기화 리스트 안에는 행 단위로 초기화할 값들을
별도의 중괄호로 명시한다.

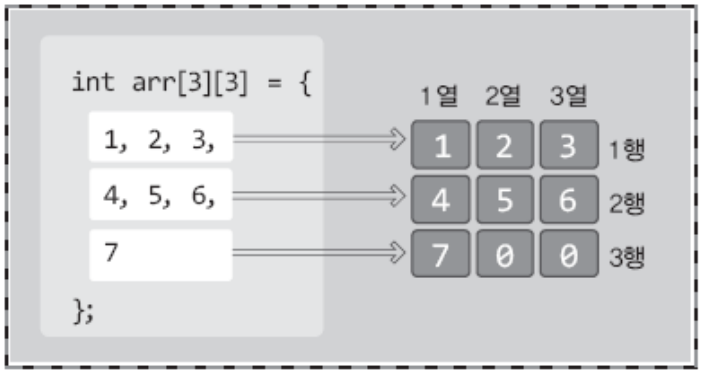




```
int arr[3][3] = {
    {1, 0, 0},
    {4, 5, 0},
    {7, 8, 9}
};
```

채워지지 않은 빈 공간은 0으로 채워진다.

2차원 배열 초기화



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};
```

초기화 리스트에 의한 배열의 크기 결정

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

두 개가 모두 비면 컴파일러가 채워 넣을 숫자를 결정하지 못한다.

8 by 1 ??

4 by 2 ??

2 by 4 ??

뒤의 것은 알려줘야 함

```
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};
```

연습 문제

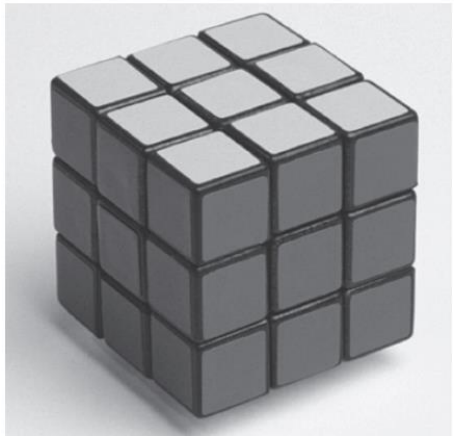
```
#include <stdio.h>
int main(void)
{
    int arr1[3][3] = { {1, 2}, 3, 4, 5, 6};

   /*
       (1) arr1에 저장된 내용은?
       (2) arr1에 저장된 내용을 2차원으로 출력할 수 있는
           프로그램 코드를 작성하세요.
   */

    return 0;
}
```

16-2 3차원! 그 이상의 배열

- 3차원 배열의 선언과 의미
  - 3차원적 메모리 구조를 의미함
  - 개념만 이해하면 충분, 일반적으로 필요 없다.

"몇 개인게 몇 개 있다" 개념 이해

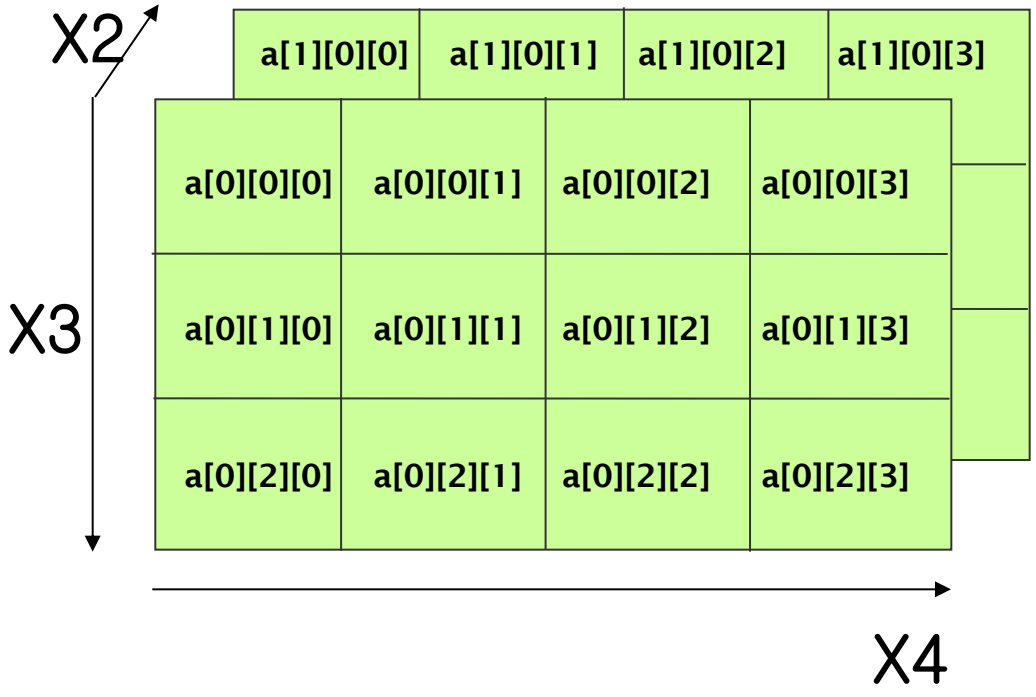
```
int a[3][3][3]
```



16-2 3차원! 그 이상의 배열

- 3차원 메모리 구조

int a[2][3][4]



a[0][0][0]
a[0][0][1]
a[0][0][2]
a[0][0][3]
a[0][1][0]
a[0][1][1]
a[0][1][2]
a[0][1][3]
a[0][2][0]
a[0][2][1]
a[0][2][2]
a[0][2][3]
a[1][0][0]
a[1][0][1]
a[1][0][2]
a[1][0][3]
a[1][1][0]
a[1][1][1]
a[1][1][2]
a[1][1][3]
a[1][2][0]
a[1][2][1]
a[1][2][2]
a[1][2][3]

3차원 배열 예제

```
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;
}
```

실행결과

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