민 선 생 코 딩 학 원 훈 련 반

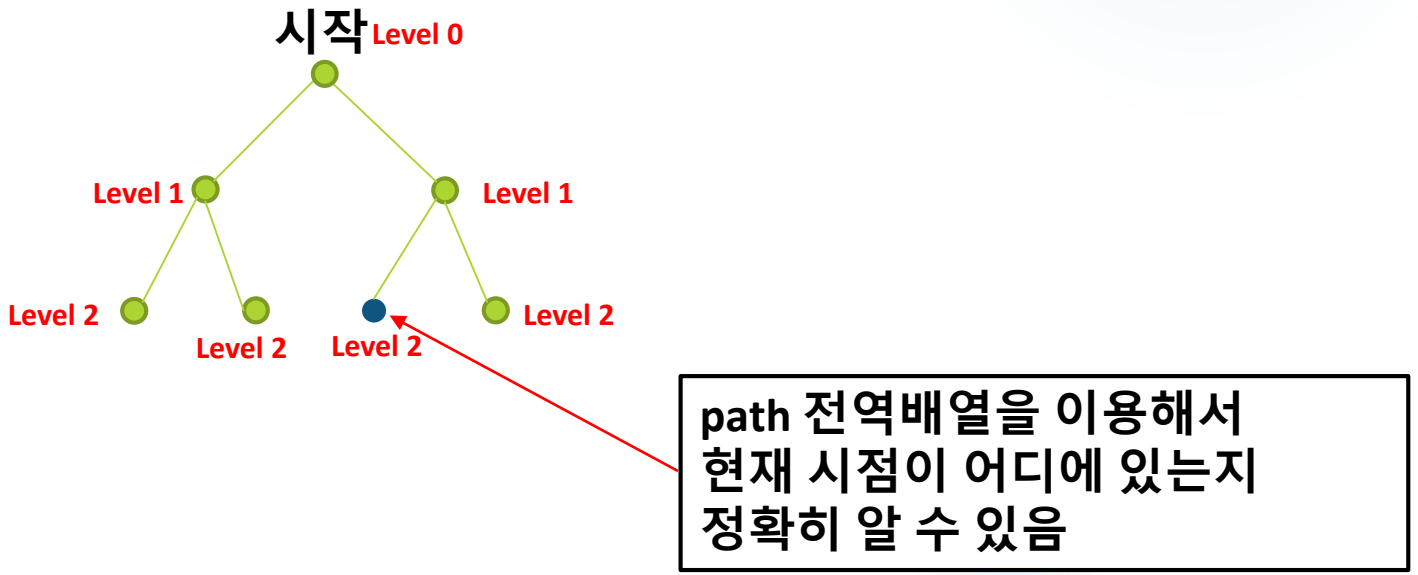
수업노트 LV-22

배우는 내용

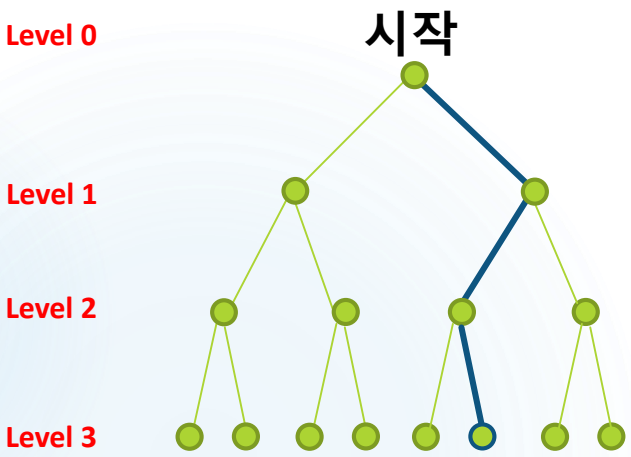
1. 재귀호출3 - path전역배열
2. 3차배열

재귀호출시 path 전역배열

- 재귀호출 할 때 내가 어느 경로로 들어와 있는지 확인하는 방법
- **전역배열에 이력을 기록하면서** 재귀호출을 진행한다.



path전역배열의 예시



- **시작 지점 부터 왼쪽으로 들어갈 때 A를 기록, 오른쪽으로 들어갈 때 B를 기록한다.**
  (안에 들어갈 값 1, 2 대신 마음대로 지정 가능)

path	B	A	B

- 만약 path배열 값이 위와 같다면
  오른쪽 > 왼쪽 > 오른쪽 으로 진입했음을 알 수 있음

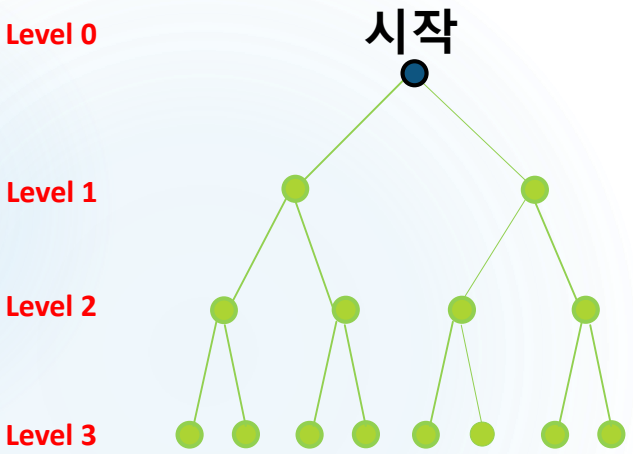
path를 사용할때 기본 규칙

- 함수호출(재귀호출)을 하기 직전
  다음 행동을 Path에 **기록을 해 주고**,
  함수호출 수행이 끝나면 **기록을 지워준다.**

1. **다음 시점을 기록**
2. **재귀호출(level + 1);**
3. **기록 지우기**

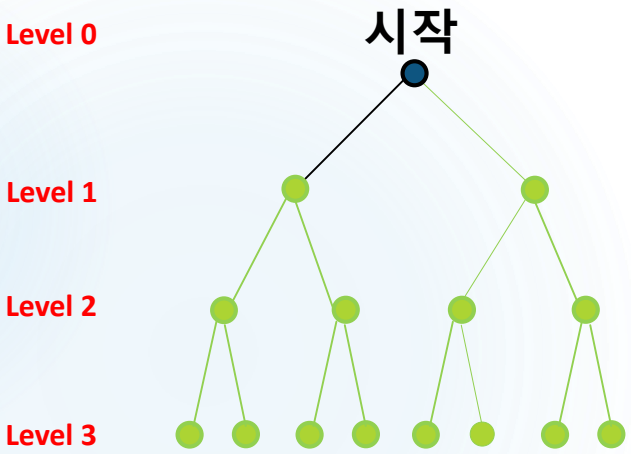
```
#include <iostream>
using namespace std;

char path[10];
void run(int level)
{
    if (level == 3)
    {
        return;
    }

    for (int x = 0; x < 2; x++)
    {
        path[level] = 'A' + x;
        run(level + 1);
        path[level] = 0;
    }
}
int main()
{
    run(0);
    return 0;
}
```

path 사용과정1



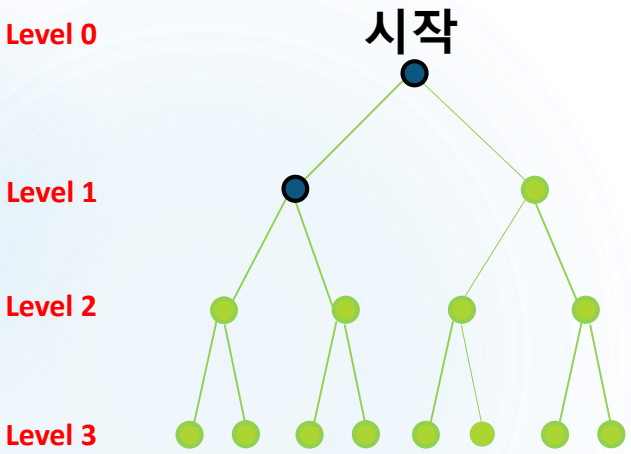
- 재귀호출 함수에 처음 진입했을 때

path 사용과정2



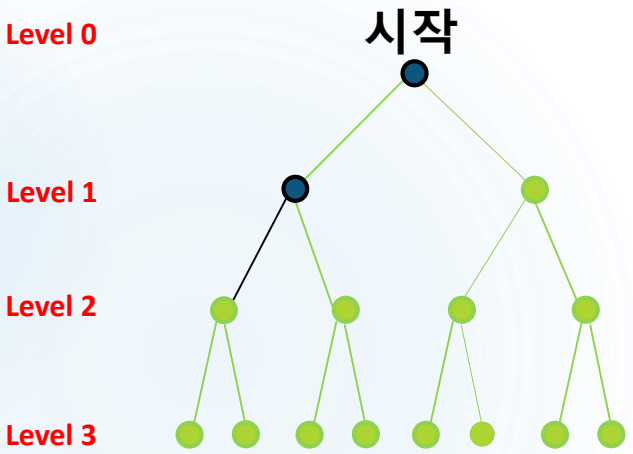
- 왼쪽으로 들어가기 직전
(이제 A로 재귀 탈 것이다.)

path 사용과정3



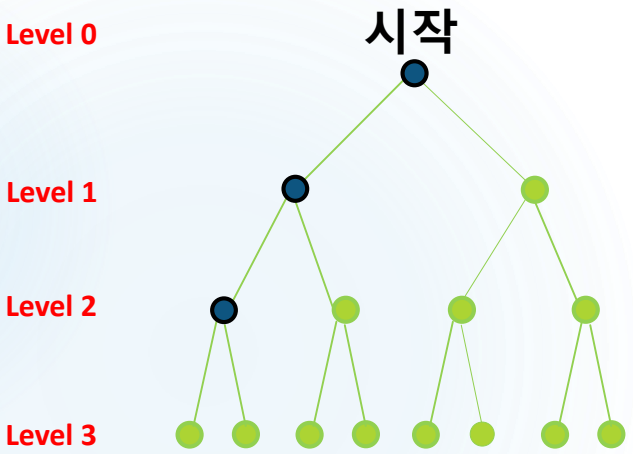
- 왼쪽 시점으로 재귀호출 진입

path 사용과정4



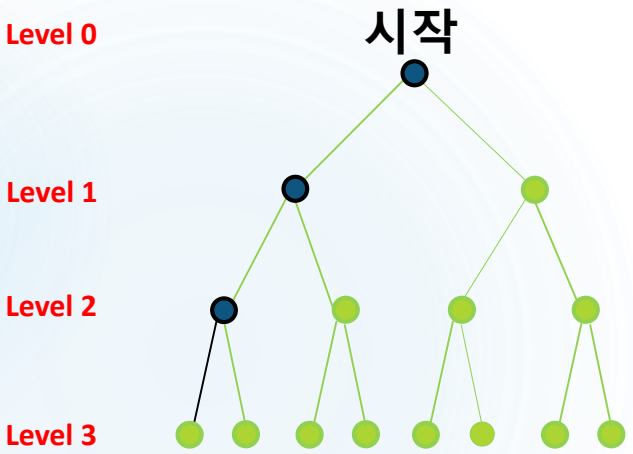
- 다시 왼쪽으로 진입 직전에 path에
  기록을 먼저 해 준다 (이제 A로 진입 할 것이다)

path 사용과정5



- Level2로 진입

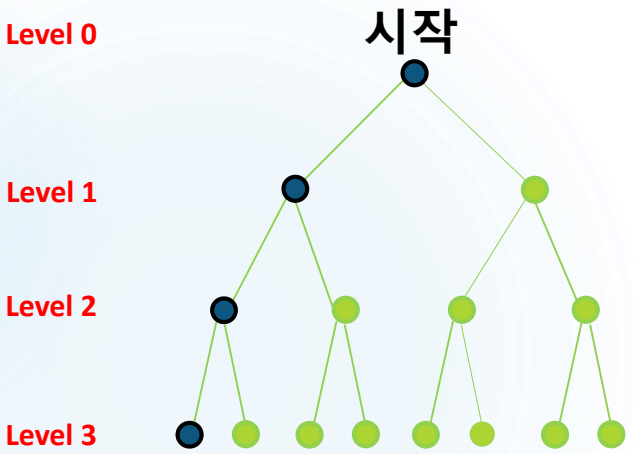
path 사용과정6



- Level3로 진입 직전
  (이제 A로 진입할 것이다)

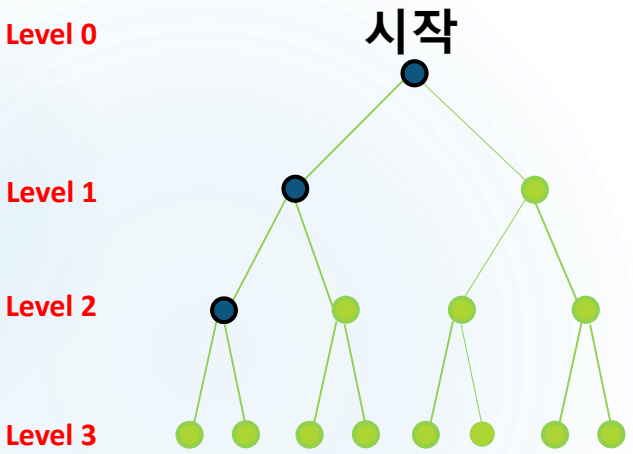
	Lev 0	Lev 1	Lev 2
path	A	A	A

path 사용과정7



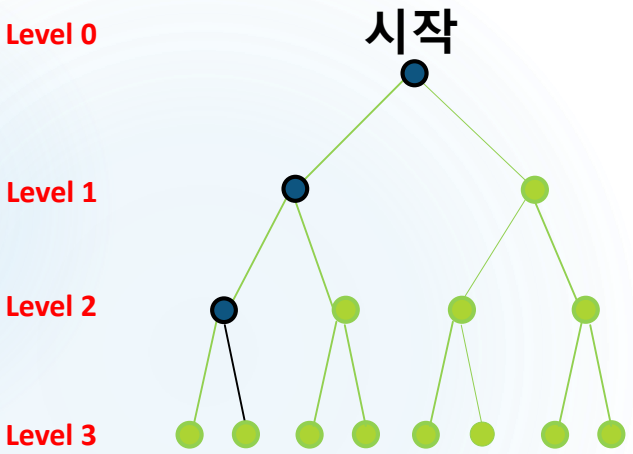
- Level3가 되었고 이제 리턴 함

path 사용과정8



- 리턴되어 Level2 시점으로 돌아 왔고,
  이제 path를 지워줌

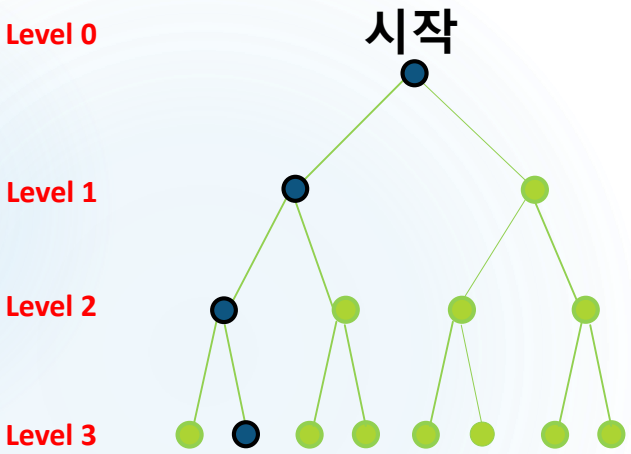
path 사용과정9



- Level3 오른쪽으로 진입 직전
(이제 B로 진입 할 것이다)

	Lev 0	Lev 1	Lev 2
path	A	A	B

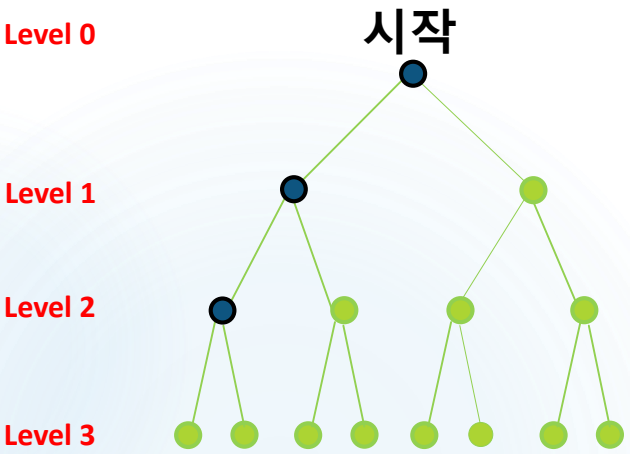
path 사용과정10



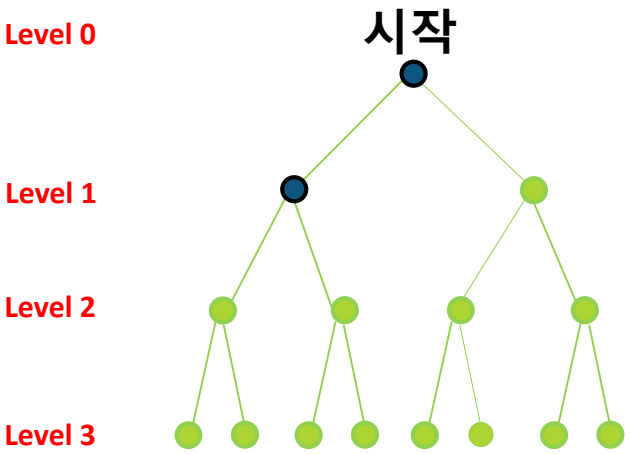
- Level3 오른쪽 진입 완료

	Lev 0	Lev 1	Lev 2
path	A	A	B

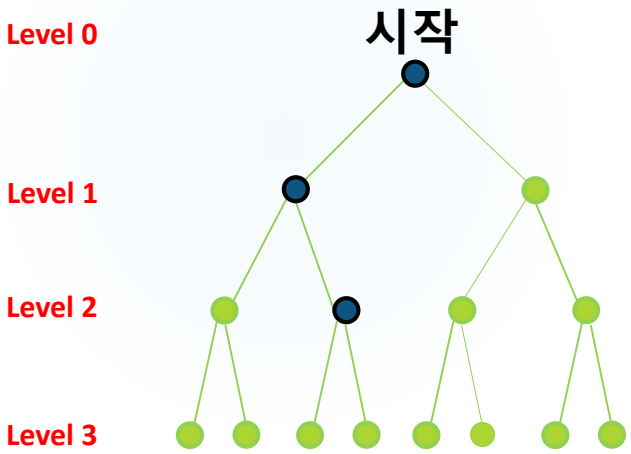
path 사용과정 계속…



	Lev 0	Lev 1	Lev 2
path	A	A	

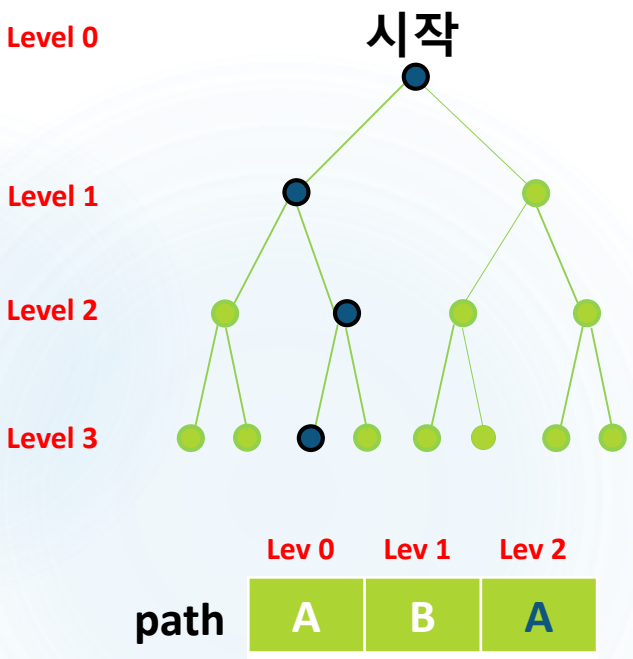


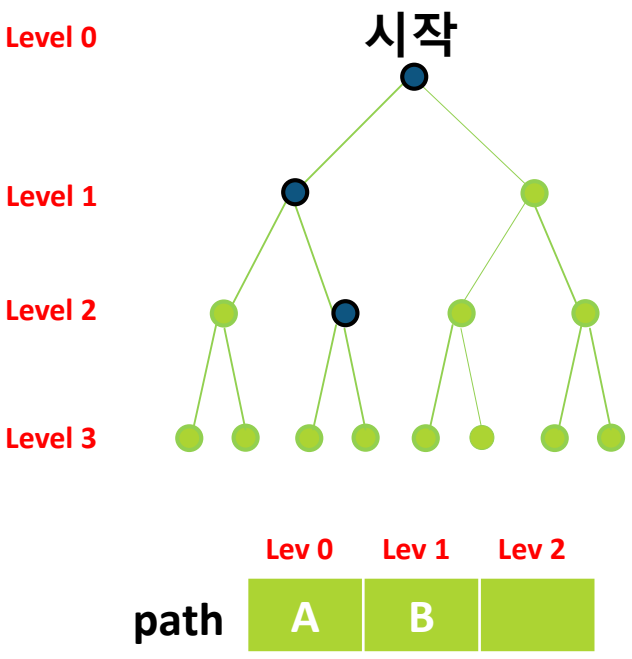
	Lev 0	Lev 1	Lev 2
path	A		

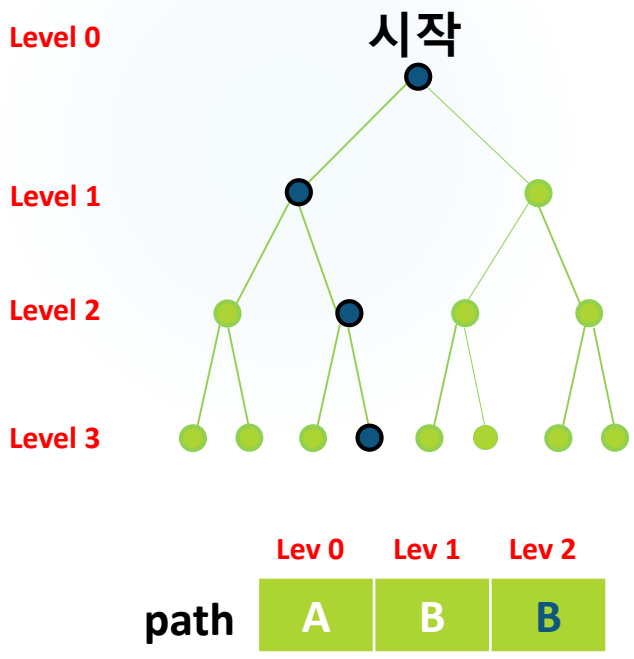


	Lev 0	Lev 1	Lev 2
path	A	B	

path 사용과정 계속…


Level 0
시작
Level 1
Level 2
Level 3
Lev 0
Lev 1
Lev 2
path
A
B
A

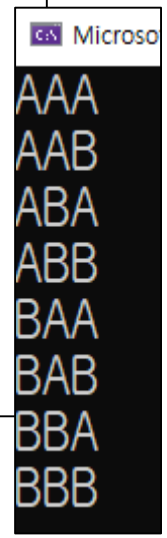

Level 0
시작
Level 1
Level 2
Level 3
Lev 0
Lev 1
Lev 2
path
A
B


Level 0
시작
Level 1
Level 2
Level 3
Lev 0
Lev 1
Lev 2
path
A
B
B

Level이 3일때 모든 경로를 출력

- Level이 3이 될 때 마다
  경로를 출력하는 소스코드

```
#include <iostream>
using namespace std;

char path[10];
void run(int level)
{
    if (level == 3)
    {
        cout << path << endl;
        return;
    }

    for (int x = 0; x < 2; x++)
    {
        path[level] = 'A' + x;
        run(level + 1);
        path[level] = 0;
    }
}
int main()
{
    run(0);
    return 0;
}
```



path에 A, B, C 대신 다른 알파벳으로 기록

- Branch가 3개, Level은 3까지 내려오는 재귀호출 코드

- name이라는 배열을 만들어서

  왼쪽으로 들어갈 때 path에 L 등록
  가운데로 들어갈때 path에 M 등록
  오른쪽으로 들어갈 때 path에 R 등록

```
#include <iostream>
using namespace std;

char path[10];
char name[4] = "LMR";

void run(int level)
{
    if (level == 3)
    {
        cout << path << endl;
        return;
    }

    for (int x = 0; x < 3; x++)
    {
        path[level] = name[x];
        run(level + 1);
        path[level] = 0;
    }
}
int main()
{
    run(0);
    return 0;
}
```

2차 배열의 특징

- 2차배열은 1차배열이 여러 개 있는 것을 나타냄

double vect[5];

5칸짜리 1차원 배열이
3개 있으면

double vect**[3]**[5];

***double Type : 소수점 저장하는 Type**

3차 배열의 이해

- 2차배열이 여러 개 있으면 3차배열

double vect[3][5];

왼쪽 2차원 배열이
2개 있으면

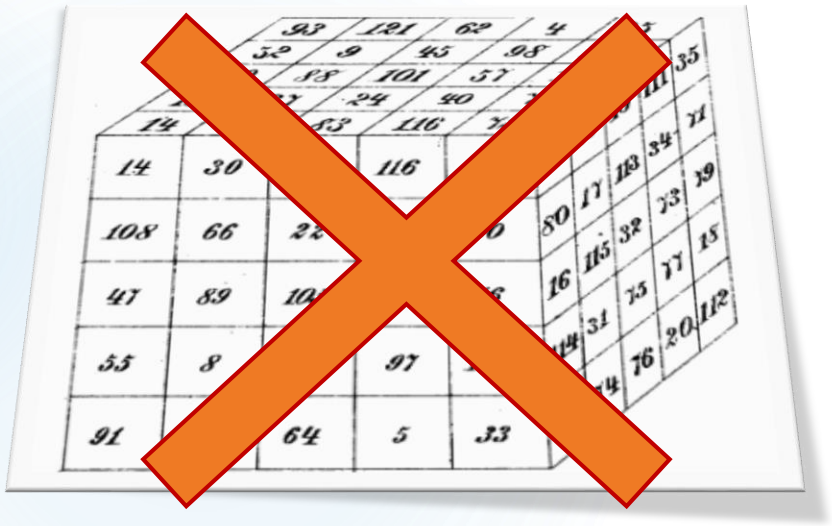
data[0][1][4] = 5;

double vect**[2]**[3][5];

vect[1][2][3] = 3;

3차배열, 4차배열, 5차배열, 6차배열…

- 3차 배열은 **3차원으로 되어있는 배열로 생각하지 말 것**
  4차 배열, 5차 배열도 존재 함



2차배열이 여러 개 있는것
➔ **3차배열**

3차 배열에 값을 채우는 예제

- 3차 배열을 입력받은 값으로 꽉 채우고 출력하는 소스코드
- 2차배열과 사용법이 동일함
- 하드코딩 하는 방법

int vect[2][3][2] = { { {2, 3}, {1, 3}, {3, 5} }, { {1, 1}, {2, 2}, {3, 3} } };

2	3
1	3
3	5

[0]

1	1
2	2
3	3

[1]

```
int vect[2][3][2] = {0};

cin >> input;

//값 채우기
for (z=0; z<2; z++)
{
   for (y=0; y<3; y++)
   {
      for (x=0; x<2; x++)
      {
         vect[z][y][x] = input;
      }
   }
}

//3차배열값 출력
for (z=0; z<2; z++)
{
   for (y=0; y<3; y++)
   {
      for (x=0; x<2; x++)
      {
         cout << vect[z][y][x];
      }
      cout << endl;
   }
   cout << endl;
}
```