# 7-1. 배열과 포인터

◈ **배열과 포인터(같은 점과 차이점)**

배열과 포인터는 메모리의 주소를 기억하게 된다.(같은 점)

---

```
char a[ ] = "KOREA";
      └──────►  배열명 a에는 배열의 시작주소가 기억된다.

char *p = "SEOUL";
      └──────► 포인터 p에는 문자열의 시작주소가 기억된다.
```

---

① 포인터와 배열명은 둘 다 주소값을 가지면서 서로 **부분적으로 호환성**도 있다.
② 그러나, 다른 점은
- **포인터** : 주소연산이 **가능**하다. (변수, p++; 가능)
- **배열명** : 주소연산이 **불가능**하다.(상수, a++; 불가)

③ 즉, 배열명은 항상 선언된 배열의 시작주소값만을 가진다.
배열명은 포인터처럼 기억된 주소값을 변경할 수 없다.
④ 배열명은 주소값을 갖는 **상수(포인터 상수)**의 개념이다.

◉ **배열과 포인터를 이용한 문자열 취급(호환성이 있다)**

---

```
void main() {
      char a[] = "KOREA";
      char *p = "SEOUL";
       printf("%c\n", *a);        //배열로 정의된 것을 포인터 방식으로 사용
      printf("%c\n", p[0]);       //포인터로 정의된 것을 배열 방식으로 사용
      printf("%s\n", a+2);
      printf("%s\n", &p[2]);      //p+2 번지부터 문자열 출력
}
```

실행결과 : K
S
REA
OUL

---

## ◈ 배열의 배열

**① C는 1차원 배열만 지원한다.**

- 정확하게 이야기하면 C에는 다차원 배열이 존재하지 않는다.
- 단지, 통상적으로 과거 다른 언어에서 표현하는 방법으로 이야기할 뿐이다.

**② 그 대신에 배열요소의 데이터 형에는 제한이 없다.**

- 이것은 '배열요소 자체가 배열인 구조로 다차원 배열구조를 구현한다'라는 뜻이다.

다음은 C의 배열 배열구조를 보여주고 있다.

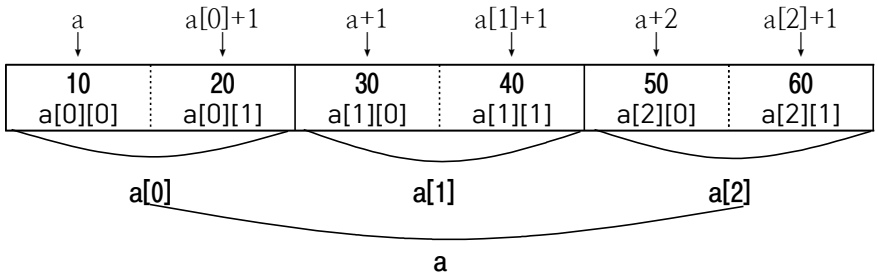
```
void main()
{
  int a[3][2] = {10,20,30,40,50,60};

  printf("%d %d %d %d \n",a[0][0],*a[0],     *(*a+0),**a );
  printf("%d %d %d %d \n",a[0][1],*(a[0]+1),*(*a+1),*((*a)+1) );
  printf("%d %d %d %d \n",a[1][0],*a[1],     *(*a+2),**(a+1) );
  printf("%d %d %d %d \n",a[1][1],*(a[1]+1),*(*a+3),*(*(a+1)+1) );
  printf("%d %d %d %d \n",a[2][0],*a[2],     *(*a+4),**(a+2) );
  printf("%d %d %d %d \n",a[2][1],*(a[2]+1),*(*a+5),*(*(a+2)+1) );
}
```

[실행결과]

```
10 10 10 10
20 20 20 20
30 30 30 30
40 40 40 40
50 50 50 50
60 60 60 60
```

- 배열 a는 배열요소 a[0], a[1], a[2]를 가지는 배열의 배열인 구조이다.



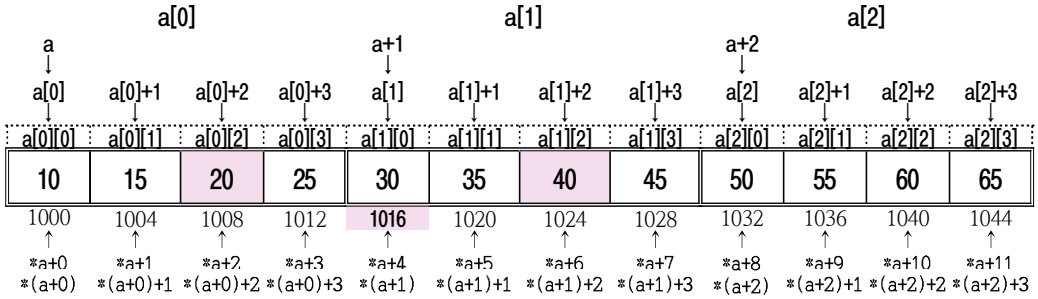
| 예제 | 다음 C 프로그램의 출력 값으로 옳은 것은? [2017년 국가 7급 유형]<br>(단, int 데이터 타입은 4byte로 표현되며, 배열 a의 시작주소는 1,000이다) |
|---|---|

```
-----------------------------------------------------------------
void main()
{
   int a[][4] = {10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65};
   printf("a+1 = %d, ", a+1);                    // 출력 : 1016
   printf("*(*a+2) = %d, ", *(*a+2));            // 출력 : 20
   printf("*(*(a+1)+2) = %d\n", *(*(a+1)+2));    // 출력 : 40
}
-----------------------------------------------------------------
```

↓

↓ 메모리 구조

↓




◇ a+1 = 1016 → int 데이터 타입의 크기는 4byte이므로

◇ *(*a+2) = 20

◇ *(*(a+1)+2) = 40

// 배열의 배열에서 내용물을 나타내는 경우

| 대괄호가 2개 있는 경우 | a[1][1] |
|---|---|
| 별이 2개 있는 경우 | *(*(a + 1) + 1) |
| 괄호와 별이 각각 1개 있는 경우 | *(a[1] + 1) |

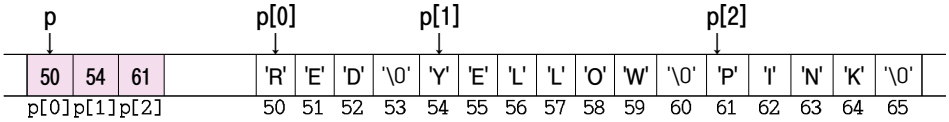
## ◈ 포인터 배열

먼저, "포인터 배열과 배열의 배열"의 차이는 다음과 같다.

---

◈ 포인터 배열

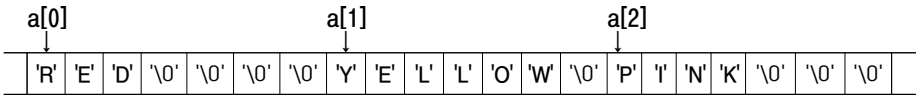
```
char *p[] = { "RED", "YELLOW", "PINK" };
```




- p는 배열의 시작주소 값을 가지고,
- p[0], p[1], p[2]는 각각 문자열이 저장된 시작주소 값을 갖는다.

◈ 배열의 배열(2차원 배열 개념)

```
char a[ ][7] = { "RED", "YELLOW", "PINK" };
```




- a와 a[0]은 같은 주소 값을 가진다.(배열명은 배열의 시작주소 값을 가지므로)
- 길이가 짧은 문자열 뒤에는 불필요한 '\0'이 많이 저장된다.(메모리 낭비)

---

① 포인터 배열은 배열요소가 메모리 주소를 기억하는 구조이다.
② 길이가 다른 문자열을 취급하는데 사용하면 메모리 낭비가 없어진다.
③ 포인터 배열은 배열과 포인터의 특징을 동시에 갖는다.
④ 포인터 배열 p와 배열의 배열 a는 둘 다 배열명이므로 주소연산이 불가능하다.

### ◉ 길이가 다른 여러 개의 문자열을 다루는 포인터 배열

---

```
void  main(){
      char *p[] = {"RED", "WHITE", "BLUE"};

      printf("%s\n", p[0]);                    //문자열 출력
      printf("%s\n", p[1]);
      printf("%s\n", p[2]);
}
```

실행결과 : RED
WHITE
BLUE

---

## 기출문제 분석

### 1. 다음 C 프로그램의 실행 결과는? [2016년 서울 9급]

```
-------------------------------------------------
#include<stdio.h>
int main() {
   char *a[2] = {"Good morning", "C language"};
   printf("%s\n", a[0]+5);
   printf("%c\n", *(a[1]+6));
   return 0;
}
-------------------------------------------------
```

① Good morning
C-language

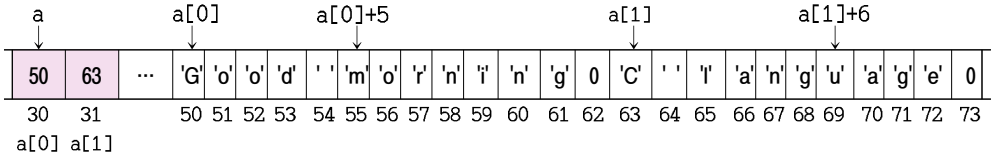
② morning
a

③ morning
g

④ morning
u

♤ 포인터 배열

• 메모리 구조는 다음과 같은 형태가 된다.(문자열 끝에 0은 null을 의미한다)
char *a[2] = {"Good morning", "C language"};




• 배열명 a는 배열의 시작주소 값을 가진다.

• printf("%s\n", a[0]+5);
→ 변환 기호가 %s이므로, a[0]+5가 가리키는 곳부터 해당 문자열 끝까지 출력
→ morning이 출력된다.

• printf("%c\n", *(a[1]+6));
→ 변환 기호가 %c이므로, a[1]+6가 가리키는 곳의 해당 문자 하나만 출력
→ u가 출력된다.

정답 : ④

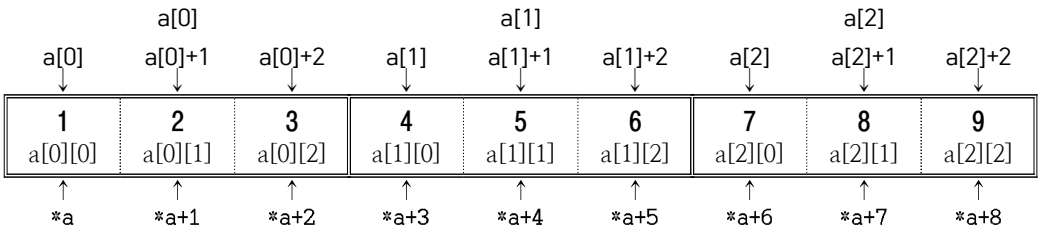
## 2. 다음 C 프로그램의 출력 값은? [2015년 지방 9급]

```
#include <stdio.h>
int main()
{
  int a[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
  int sum1, sum2;
  sum1 = *(*a + 1) + *(*a + 2);
  sum2 = *a[1] + *a[2];
  printf("%d, %d", sum1, sum2);
}
```

① 3, 5 ② 5, 5
③ 5, 11 ④ 11, 5

### ♤ 배열의 배열(2차원 배열)

• 배열구조를 그림으로 나타내면 다음과 같다.




sum1 = *(*a + 1) + *(*a + 2) = 2 + 3 = **5**

sum2 = *a[1] + *a[2] = 4 + 7 = **11**

정답 : ③